

LIS User's Guide

October 21, 2005

Revision 4.1

History:

Revision	Summary of Changes	Date
4.1	LIS version 4.0.2 release	October 21, 2005
4.0	Milestone "K" submission	February 11, 2005
3.2	LIS version 3.1 release	December 17, 2004
3.1	Milestone "G" release	July 16, 2004
3.0	Milestone "G" submission	May 7, 2004
2.3	Improvements to Milestone "I"	November 30, 2003
2.2	-	-
2.1	Milestone "I" release	November 10, 2003
2.0	Milestone "I" submission	August 14, 2003
1.1	Milestone "F" release	April 25, 2003
1.0	Milestone "F" submission	March 31, 2003



National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Contents

1	Introduction	4
1.1	What's New	4
1.1.1	LIS 4.0 – 4.0.2	4
1.1.2	LIS 3.1 – 4.0	4
1.1.3	LIS 3.0 – 3.1	5
1.1.4	LIS 2.0 – 3.0	5
2	Background	6
2.1	LIS	6
2.2	LIS driver	6
2.3	Community Land Model (CLM)	7
2.4	The Community Noah Land Surface Model	8
2.5	Variable Infiltration Capacity (VIC) Model	8
2.6	GrADS-DODS Server	9
3	Preliminaries	10
4	Running Modes	11
4.1	Single-Process-Based Running Mode	11
4.2	MPI-Based Running Mode	11
4.3	GDS-Based Running Mode	11
4.4	1 km Global Runs	12
5	Obtaining the Source Code	13
5.1	Downloading the Source Code	13
5.2	Source files	13
5.3	Scripts	15
5.4	Post-processing	15
5.5	Opendap Scripts	16
6	Obtaining The Data-sets	17
6.1	Downloading The GSWP-2 Data-sets	17
6.2	Downloading The LIS Sample Data-sets	17
6.2.1	Downloading Sample Parameter Data-sets	17
6.2.2	Example: Downloading Sample 1/4 Deg. Parameter Data-sets Via HTTP	17
6.2.3	Example: Downloading Sample 1/4 Deg. Parameter Data-sets Via GDS	20
6.2.4	Downloading Sample Forcing Data-sets	22
6.2.5	Example: Downloading Sample 1/4 Deg. Forcing Data-sets Via HTTP	22
6.3	Downloading Sample Output Data-sets	23
6.3.1	Example: Downloading Sample 1/4 Deg. Output Data-sets Via GDS	23

6.3.2	Viewing The Sample 1/4 Deg. Output Data-sets	25
7	Building the Executable	26
7.1	Compiling In GSWP-2 Support	26
7.2	Compiling In MPI Support	27
7.3	General Build Instructions	27
7.3.1	Required Software Libraries	27
7.3.2	Notes on compiling MPICH	28
7.3.3	Modifying the Makefile	29
7.4	Compiling GrADS-DODS Support	29
7.5	Generating documentation	30
8	Running The Executable	31
8.1	Configuring Run Via LIS Card File	31
8.1.1	driver namelist	31
8.1.2	lis_run_inputs namelist	32
8.1.3	run_domain namelist	37
8.1.4	param_domain namelist	37
8.1.5	landcover namelist	38
8.1.6	elevation namelist	38
8.1.7	soils namelist	38
8.1.8	lai namelist	40
8.1.9	gswp namelist	40
8.1.10	geos namelist	41
8.1.11	gdas namelist	41
8.1.12	nldas namelist	41
8.1.13	ecmwf namelist	42
8.1.14	berg namelist	42
8.1.15	cmap namelist	42
8.1.16	agrmet namelist	43
8.1.17	clm2 namelist	43
8.1.18	noah namelist	43
8.1.19	vic namelist	45
8.1.20	opendap namelist	46
8.2	Domain Example	46
8.3	Running Over The 1 km Domain	48
9	Output Data Processing	52
9.1	CLM Output	54
9.2	Noah Output	57
9.3	VIC Output	59
A	LIS Card File	61
B	Makefile	68

1 Introduction

This is the LIS' User's Guide. This document describes how to download and install the code and data needed to run the LIS executable for LIS' revision 4.0.2. It describes how to build and run the code, and finally this document also describes how to download output data-sets to use for validation. Updates to this document will provide more detailed instructions on how to configure the executable and will address the graphical user interface.

This document consists of 9 sections, described as follows:

- 1 Introduction:** the section you are currently reading
- 2 Background:** general information about the LIS project
- 3 Preliminaries:** general information, steps, instructions, and definitions used throughout the rest of this document
- 4 Running modes:** different parallel running modes of operation
- 5 Obtaining the Source Code:** the steps needed to download the source code
- 6 Obtaining the Data-sets:** the steps needed to download the data-sets
- 7 Building the Executable:** the steps needed to build the LIS executable
- 8 Running the Executable:** the steps needed to prepare and submit a run, also describes the various run-time configurations
- 9 Output Data Processing:** the steps needed to post-process generated output for visualization

1.1 What's New

See *RELEASE_NOTES* found in the *source.tar.gz* file for more details. (See Section 5.)

1.1.1 LIS 4.0 – 4.0.2

- 1. GSWP-2 support – LIS can now run GSWP-2 experiments. Currently only CLM and Noah models have support. This support is still under development, and it should be considered beta.
- 2. Updates to the 1km running mode.
- 3. Updates to the GDS running mode.

1.1.2 LIS 3.1 – 4.0

- 1. VIC 4.0.5 – LIS' implementation of VIC has been reinstated.

1.1.3 LIS 3.0 – 3.1

1. New domain-plugin support – facilitates creating new domains.
2. New domain definition support – facilitates defining running domains. Sub-domain selection now works for both MPI-based and non MPI-based runs.
3. New parameter-plugin support – facilitates adding new input parameter data-sets.
4. New LIS version of ipolates – facilitates creating new domains and base forcing data-sets.
5. Compile-time MPI support – MPI libraries are no longer required to compile LIS.
6. Compile-time netCDF support – netCDF libraries are no longer required to compile LIS.
7. New LIS time manager support – ESMF time manager was removed. ESMF libraries are not required in this version of LIS.

1.1.4 LIS 2.0 – 3.0

1. Running Modes – Now there is more than one way to run LIS. In addition to the standard MPI running mode, there are the GDS running mode and the 1 km running mode. See Section 4 for more details.
2. Sub-domain Selection – Now you are no longer limited to global simulations. You may choose any sub-set of the global domain to run over. See Section 8.1.3 and Section 8.1.4 for more details. (This is currently only available for the MPI-based running mode.)
3. Plug-ins – Now it is easy to add new LSM and forcing data-sets into the LIS driver. See LIS’ Developer’s Guide for more details.

2 Background

This section provides some general information about the LIS project and land surface modeling.

2.1 LIS

The primary goal of the LIS project is to build a system that is capable of performing high resolution land surface modeling at high performance using scalable computing technologies. The LIS software system consists of a number of components: (1) LIS driver: the core software that integrates the use of land surface models, data management techniques, and high performance computing. (2) community land surface models such as CLM [4], Noah [6], and VIC [7], and (3) Visualization and data management tools such as GrADS [1] -DODS [5] server. One of the important design goals of LIS is to develop an interoperable system to interface and interoperate with land surface modeling community and other earth system models. LIS is designed using an object oriented, component-based style. The adaptable interfaces in LIS can be used by the developers to ease the cost of development and foster rapid prototyping and development of applications. The following sections describe the main components of LIS.

2.2 LIS driver

The core of LIS software system is the LIS driver that controls program execution. The LIS driver is a model control and input/output system (consisting of a number of subroutines, modules written in Fortran 90 source code) that drives multiple offline one-dimensional LSMs. The one-dimensional LSMs such as CLM and Noah, apply the governing equations of the physical processes of the soil-vegetation-snowpack medium. These land surface models aim to characterize the transfer of mass, energy, and momentum between a vegetated surface and the atmosphere. When there are multiple vegetation types inside a grid box, the grid box is further divided into “tiles”, with each tile representing a specific vegetation type within the grid box, in order to simulate sub-grid scale variability.

The execution of the LIS driver starts with reading in the user specifications, including the modeling domain, spatial resolution, duration of the run, etc. Section 8 describes the exhaustive list of parameters specified by the user. This is followed by the reading and computing of model parameters. The time loop begins and forcing data is read, time/space interpolation is computed and modified as necessary. Forcing data is used to specify the boundary conditions to the land surface model. The LIS driver applies time/space interpolation to convert the forcing data to the appropriate resolution required by the model. The selected model is run for a vector of “tiles” and output and restart files are written at the specified output interval.

Some of the salient features provided by the LIS driver include:

- Vegetation type-based “tile” or “patch” approach to simulate sub-grid scale variability.
- Makes use of various satellite and ground-based observational systems.
- Derives model parameters from existing topography, vegetation, and soil coverages.
- Extensible interfaces to facilitate incorporation of new land surface models, forcing schemes.
- Uses a modular, object oriented style design that allows “plug and play” of different features by allowing user to select only the components of interest while building the executable.
- Ability to perform regional modeling (only on the domain of interest).
- Provides a number of scalable parallel processing modes of operation.

Please refer to the software design document for a detailed description of the design of LIS driver. The LIS developer’s guide describes how to use the extensible interfaces in LIS. The “plug and play” feature of different components is described in this document.

2.3 Community Land Model (CLM)

CLM (Community Land Model) is a 1-D land surface model, written in Fortran 90, developed by a grass-roots collaboration of scientists who have an interest in making a general land model available for public use. LIS currently uses CLM version 2.0. CLM version 2.0 was released in May 2002. The source code for CLM 2.0 is freely available from the National Center for Atmospheric Research (NCAR) [4]. The CLM is used as the land model for the Community Climate System Model (CCSM) (<http://www.cesm.ucar.edu/>), which includes the Community Atmosphere Model (CAM) (<http://www.cgd.ucar.edu/cms/>). CLM is executed with all forcing, parameters, dimensioning, output routines, and coupling performed by an external driver of the user’s design (in this case done by LIS). CLM requires pre-processed data such as the land surface type, soil and vegetation parameters, model initialization, and atmospheric boundary conditions as input. The model applies finite-difference spatial discretization methods and a fully implicit time-integration scheme to numerically integrate the governing equations. The model subroutines apply the governing equations of the physical processes of the soil-vegetation-snowpack medium, including the surface energy balance equation, Richards’ [12] equation for soil hydraulics, the diffusion equation for soil heat transfer, the energy-mass balance equation for the snowpack, and the Collatz et al. [9] formulation for the conductance of canopy transpiration.

2.4 The Community Noah Land Surface Model

The community Noah Land Surface Model is a stand-alone, uncoupled, 1-D column model freely available at the National Centers for Environmental Prediction (NCEP; [6]). The name is an acronym representing the various developers of the model (N: NCEP; O: Oregon State University, Dept. of Atmospheric Sciences; A: Air Force (both AFWA and AFRL - formerly AFGL, PL); and H: Hydrologic Research Lab - NWS (now Office of Hydrologic Development - OHD)). Noah can be executed in either coupled or uncoupled mode. It has been coupled with the operational NCEP mesoscale Eta model [10] and its companion Eta Data Assimilation System (EDAS) [13], and the NCEP Global Forecast System (GFS) and its companion Global Data Assimilation System (GDAS). When Noah is executed in uncoupled mode, near-surface atmospheric forcing data (e.g., precipitation, radiation, wind speed, temperature, humidity) is required as input. Noah simulates soil moisture (both liquid and frozen), soil temperature, skin temperature, snowpack depth, snowpack water equivalent, canopy water content, and the energy flux and water flux terms of the surface energy balance and surface water balance. The model applies finite-difference spatial discretization methods and a Crank-Nicholson time-integration scheme to numerically integrate the governing equations of the physical processes of the soil vegetation-snowpack medium, including the surface energy balance equation, Richards' [12] equation for soil hydraulics, the diffusion equation for soil heat transfer, the energy-mass balance equation for the snowpack, and the Jarvis [11] equation for the conductance of canopy transpiration.

2.5 Variable Infiltration Capacity (VIC) Model

Variable Infiltration Capacity (VIC) model is a macroscale hydrologic model, written in C, being developed at the University of Washington and Princeton University. The VIC code repository along with the model description and source code documentation is publicly available at the Princeton website [7]. VIC is used in macroscopic land use models such as SEA - BASINS (<http://boto.ocean.washington.edu/seasia/intro.htm>). VIC is a semi-distributed, grid-based hydrological model, which parameterizes the dominant hydrometeorological processes taking place at the land surface - atmospheric interface. The execution of VIC model requires preprocessed data such as precipitation, temperature, meteorological forcing, soil and vegetation parameters, etc. as input. The model uses three soil layers and one vegetation layer with energy and moisture fluxes exchanged between the layers. The VIC model represents surface and subsurface hydrologic processes on a spatially distributed (grid cell) basis. Partitioning grid cell areas to different vegetation classes can approximate sub-grid scale variation in vegetation characteristics. VIC models the processes governing the flux and storage of water and heat in each cell-sized system of vegetation and soil structure. The water balance portion of VIC is based on three concepts: 1) Division of grid-cell into fraction sub-grid vegetation coverages.

- 2) The variable infiltration curve for rainfall/runoff partitioning at the land surface.
- 3) A baseflow/deep soil moisture curve for lateral baseflow.

Water balance calculations are preformed at three soil layers and within a vegetation canopy. An energy balance is calculated at the land surface. A full description of algorithms in VIC can be found in the references listed at the VIC website.

2.6 GrADS-DODS Server

A GrADS-DODS Server (GDS) is a data server built upon the Grid Analysis and Display System (GrADS) and the Distributed Oceanographic Data System (DODS).

GrADS is an earth science data manipulation and visualization tool under development at the Center for Ocean-Land-Atmosphere Studies (COLA) (<http://http://grads.iges.org/cola.html>). See <http://grads.iges.org/grads/grads.html> for more detailed information about GrADS.

DODS, also called the Open source Project for a Network Data Access Protocol (OPeNDAP), is a protocol for serving data-sets stored in various formats over a network. See <http://www.unidata.ucar.edu/packages/dods/> for more detailed information about DODS.

A GDS may be used to provide the LIS driver with the forcing and input parameter data needed to run an LSM.

A GDS is an optional component of the LIS system. LIS may be run without using a GDS to access the forcing and input parameter data-sets. All necessary forcing and input parameter data-sets may be stored on locally-accessable hard-disks and read in directly by the LIS driver, provided the computer system has sufficient memory.

The intent of a GDS for the LIS project is to provide the LIS driver with subsets of the forcing and input parameter data-sets, so that large-scale, high-resolution domains may be broken-up/parallelized and processed across many compute-nodes of a Beowulf cluster.

3 Preliminaries

This code has been compiled and run on Linux PC (Intel/AMD based) systems, IBM AIX systems, SGI IRIX64 6.5 systems, and SGI Altix system. These instructions expect that you are using such a system. In particular you need Software:

- Linux
 - Absoft's Pro Fortran Software Developement Kit, version 8.0
or
Lahey/Fujitsu's Fortran 95 Compiler, release L6.00c
 - GNU's C and C++ compilers, gcc and g++, version 2.96
 - MPICH , version 1.2.5.2
 - GNU's make, gmake, version 3.77
- IBM
 - XL Fortran version 8 release 1
 - GNU's make, gmake, version 3.77
- SGI IRIX
 - MIPSpro version 7.3.1.1m
 - Message Passing Toolkit, mpt, version 1.5.3.0
 - GNU's make, gmake, version 3.77
- SGI Altix
 - Intel Fortran Compiler version 7
 - Message Passing Toolkit, mpt, version 1.9.1.0
 - GNU's make, gmake, version 3.77

System Resources:

	1/4 deg	5 km	1 km
memory	250 MB	32 GB	800 MB (per patch)
hard disk space	3 GB	46 GB	850 GB
source	64 MB	64 MB	64 MB
input data	1.5 GB	25 GB	200 GB
output data	1 GB	20 GB	640 GB

Note, the requirements for input and output data are for a 1-day simulation.

You need to create a working directory on your system that has sufficient disk space to install and run in. Throughout the rest of this document this directory shall be referred to as *\$WORKING*.

4 Running Modes

The computational and resource requirements increase significantly for global modeling at high resolutions such as 5 km and 1 km. The land surface modeling component in LIS is designed to handle these requirements and perform high-performance, parallel simulation of global, regional, and local land surface processes with a number of land surface models.

LIS is designed to operate in a number of high performance running modes to meet the diverse requirements of distributed memory and shared memory platforms. LIS can operate in two different parallel modes based on the way data is handled by the LIS driver. In the message passing interface (MPI)-based paradigm, a master processor handles the data for the entire domain, computes the domain decomposition, and subsequently distributes the data onto the compute nodes. This paradigm is limited by the amount of memory available to the master processor. On a shared memory platform, a pool of processors can be used to make a large amount of memory available. To handle increased memory requirements and the limited resources available on a distributed memory environment, a GrADS-DODS Server (GDS)-based running mode can be used in LIS. In this mode of operation, the compute nodes retrieve data from a GDS. This mode of operation is no longer constrained by the lack of a large pool of memory on the master processor.

The LIS driver also includes the capability to perform regional modeling in addition to global scales. The domain information is specified by the user, and the LIS driver handles the subsetting tasks. In the MPI-mode, the subsetting information is derived from a larger domain, whereas in the GDS-mode, the subsetting is carried out by requesting appropriate data from the GDS-server. The details of using these different options are described in the following sections.

Note, *both* parallel running modes require the Message Passing Interface libraries. If a single-process, i.e. non-parallel, version of LIS is desired, LIS may be compiled without the MPI libraries.

4.1 Single-Process-Based Running Mode

LIS' default running mode is a non-parallel running mode. Simply follow the directions in this document to run LIS in this mode.

4.2 MPI-Based Running Mode

In order to run LIS using MPI, you must first install the MPI libraries onto your system. Then follow the extra instructions in Section 7 to compile in MPI support.

4.3 GDS-Based Running Mode

In order to run LIS using the GDS-based running mode, you must first install a GDS and a DODS enabled version of GrADS. Then you must compile the

GDS-based running mode support into LIS' executable.

To install a GrADS-DODS Server simply go to <http://grads.iges.org/grads/gds/> and follow the on-line instructions.

Once you have installed the GDS, you must install a DODS enabled version of GrADS. Go to <http://grads.iges.org/grads/grads.html> and follow the downloading instructions.

After you have installed both the GDS and the GrADS packages, you must edit the GDS data retrieving script, *\$WORKING/LIS/.opendap_scripts/getdata.pl*. You must modify the definitions of the *\$server* and *\$GrADS* variables to reflect your installation.

See Section 5 for instructions on downloading the source code, and see Section 7.4 for instructions on compiling the GDS-based running mode support into LIS' executable.

Note, currently, the GDS-based running mode only supports global simulations.

4.4 1 km Global Runs

The 1 km global run is a special case of the GDS-based running mode. You must first follow the steps in Section 4.3 to properly configure your system. Once configured, you must run the special 1 km scripts.

The main 1 km script drives a pool-of-tasks scheme for parallelizing the computations over the global 1 km domain. This global domain has already been divided into 1183 sub-domain patches, each containing 720×300 grid-points. The driver script monitors the availability of each compute node of LIS' Linux cluster, and it pushes a sub-domain patch onto each free node. Should, for any reason, the computations on a compute node crash, the corresponding sub-domain patch is returned to the list of patches-to-complete (the "pool"), where it will wait until it is reassigned.

5 Obtaining the Source Code

This section describes how to obtain the source code needed to build the LIS executable.

5.1 Downloading the Source Code

To obtain the source code needed for LIS' revision 4.0.2:

1. Go to LIS' "Public Release Home Page"
Go to <http://lis.gsfc.nasa.gov/>
Follow the "Source Codes" link.
Follow the "LIS 4.0.2 Code Release" link.
2. From LIS' "Public Release Home Page"
Follow the "LIS 4.0.2 Source Code and Scripts" link.
3. Download the required *source.tar.gz* and *scripts.tar.gz* files into your working directory, *\$WORKING/LIS*.
Use the "LIS driver and Land Surface Models" link to get *source.tar.gz*.
Use the "LIS scripts" link to get *scripts.tar.gz*.
4. Unpack these files. Run (in the order listed):

```
% gzip -dc source.tar.gz | tar xf -  
% gzip -dc scripts.tar.gz | tar xf -
```

Unpacking the *scripts.tar.gz* file will also create the input directory tree needed for downloading the input data-sets.

5. Download the optional (if desired) *postproc.tar.gz* and *opendap_scripts.tar.gz* files into your working directory, *\$WORKING/LIS*.
Use the "Post-processing scripts" link to get *postproc.tar.gz*.
Use the "GDS scripts" link to get *opendap_scripts.tar.gz*.
Unpack them.

```
% gzip -dc postproc.tar.gz | tar xf -  
% gzip -dc opendap_scripts.tar.gz | tar xf -
```

5.2 Source files

Unpacking the *source.tar.gz* file will create a *\$WORKING/LIS/src* sub-directory. The structure of *src* is as follows:

Directory Name	Synopsis
baseforcing	Top level directory for base forcing methods
baseforcing/berg	Routines for handling ECMWF reanalysis forcing product
baseforcing/ecmwf	Routines for handling ECMWF forcing product
baseforcing/gdas	Routines for handling GDAS forcing product
baseforcing/gdas-ncep	Routines for handling GDAS forcing product (NCEP port)
baseforcing/geos	Routines for handling GEOS forcing product
baseforcing/gswp	Routines for handling GSWP-2 forcing product
baseforcing/nldas	Routines for handling NLDAS forcing product
core	LIS core routines
domain-plugin	Modules defining the function table registry of included domain types
domains	Routines for creating tiles
forcing-plugin	Modules defining function table registries of included model forcing, observed radiation, and precipitation forcing products.
interp	Interpolation routines (Adopted from NCEP's ipolates library)
lib	Libraries needed for linking
lsm-plugin	Modules defining the function table registry of included LSMs
lsms/clm2	Top level clm2 land surface model sub-directory
lsms/clm2/biogeochem	Biogeochemistry routines
lsms/clm2/biogeophys	Biogeophysics routines (e.g., surface fluxes)
lsms/clm2/camclm_share	Code shared between the clm2 and cam (e.g., calendar information)
lsms/clm2/csm_share	Code shared by all the geophysical model components of the Community Climate System Model (CCSM). Currently contains code for CCSM message passing orbital calculations and system utilities
lsms/clm2/ecosysdyn	Ecosystem dynamics routines (e.g., leaf and stem area index)
lsms/clm2/main	Control (driver) routines
lsms/clm2/mksrfddata	Routines for generating surface data-sets
lsms/clm2/utills	Independent utility routines
lsms/hyssib	hyssib land surface model
lsms/mosaic	mosaic land surface model
lsms/noah.2.6	Noah land surface model version 2.6
lsms/noah.2.6-ncep	Noah land surface model version 2.6 (NCEP port)
lsms/noah.2.7	Noah land surface model version 2.7.1
lsms/ssib	ssib land surface model
lsms/template	support for running LIS without an LSM
lsms/vic	VIC land surface model

make	Makefile and needed headers
obsprecips	Top level directory for observed precipitation products
obsprecips/cmap	Routines for handling CMAP precipitation product
obsprecips/huff	Routines for handling HUFFMAN precipitation product
obsprecips/pers	Routines for handling PERSIANN precipitation product
obsrads	Top level directory for observed radiation products
obsrads/agrmet	Routines for handling AGRMET radiation product
param-plugin	Modules defining the function table registry of included input parameter reading routines
params	Top level directory for input parameter products
params/elev	Routines for handling elevation correction data
params/lai	Routines for handling leaf/stem area index data
params/landcover	Routines for handling land cover classification data
params/soils	Routines for handling soil classification data
tables	Contains the GRIB tables for writing grib output
w3lib	Contains the new GRIB library source code from NCEP

Source code documentation may be found on LIS' web-site on LIS' "Public Release Home Page". Follow the "LIS Source Code Documentation" link.

5.3 Scripts

The *scripts.tar.gz* file contains a script for compiling and building the executable and a sample card file used for running the LIS executable and for configuring the individual runs. These are described in Section 8.

Unpacking the *scripts.tar.gz* file will place the following files into the *\$WORKING/LIS* sub-directory:

File Name	Synopsis
lis.crd	Sample card file
comp.csh	Compile and build script
utils	The in-line documentation processing scripts
input	An empty directory tree to hold the input data

5.4 Post-processing

The *postproc.tar.gz* file contains the source and data files needed for generating data-plots using GrADS [1]. Post-processing is described in Section 9.

Unpacking the *postproc.tar.gz* file will create a *\$WORKING/LIS/postproc* sub-directory. The structure of *postproc* is as follows:

File Name	Synopsis
noah.25ctl	GrADS descriptor file for 1/4 deg. Noah data
noah.25.gs	Script for generating 1/4 deg. Noah data plots
clm.25ctl	GrADS descriptor file for 1/4 deg. CLM data
clm.25.gs	Script for generating 1/4 deg. CLM data plots
vic.25ctl	GrADS descriptor file for 1/4 deg. VIC data
vic.25.gs	Script for generating 1/4 deg. VIC data plots
noah.gswpctl	GrADS descriptor file for GSWP-2 Noah data
clm.gswpctl	GrADS descriptor file for GSWP-2 CLM data
pdef	Directory containing global pdef files (only needed when reading 1-d output data files or GSWP-2 data)

5.5 Opendap Scripts

The *opendap_scripts.tar.gz* file contains the scripts used by LIS to access data from a GDS server.

Unpacking the *opendap_scripts.tar.gz* file will create a *\$WORKING/LIS/opendap_scripts* sub-directory. The structure of *opendap_scripts* is as follows:

File Name	Synopsis
getdata.pl	Main GDS data-retrieval script
links.sh	Symbolic link generating script

The *getdata.pl* Perl script retrieves data from a GDS by issuing GrADS commands. See the documentation for *getdata.pl* for more details about this script.

Note, you must modify the definitions of the **\$server** and **\$GrADS** variables to reflect your installation of a GDS and GrADS. See Section 4.3.

LIS never directly calls the *getdata.pl* script by name, rather it calls the script by one of the script's many aliases. These aliases (or symbolic links) must be created before LIS is run. Run the *links.sh* script to generate all the links needed by LIS driver.

6 Obtaining The Data-sets

This section describes how to obtain the various data-sets for testing the LIS executable.

6.1 Downloading The GSWP-2 Data-sets

Note, GSWP-2 support in this release of LIS (4.0.2) should be considered beta.

The preferred method for testing LIS is to perform a GSWP-2 baseline run.

Please see the GSWP web-site (<http://www.iges.org/gswp/>) for details on how to download the data-sets and on how to perform the baseline run.

6.2 Downloading The LIS Sample Data-sets

The old method for testing LIS was to use sample data-sets provided by the LIS project.

To obtain these sample data-sets for LIS' revision 4.0.2:

1. Go to LIS' "Home Page"

Go to <http://lis.gsfc.nasa.gov/>

2. Follow the "Get Sample Data" link.

The "LIS 4.0" Section provides links to land surface parameters and atmospheric forcing data.

6.2.1 Downloading Sample Parameter Data-sets

Land surface models simulate the physical and dynamical processes of the land surface. Driven by external forcing, the spatial and temporal evolution of these processes are intrinsically determined by the physical and dynamical properties, or parameters, of the land surface. Please follow the link to land surface parameters under "LIS 4.0" to obtain parameter data-sets. It is recommended that the files be organized according to the domain resolution and land surface model type.

The following section provide examples of how to download sample 1/4 deg. parameter data-sets. Downloading sample 5 km and 1 km parameter data-sets is similar.

6.2.2 Example: Downloading Sample 1/4 Deg. Parameter Data-sets Via HTTP

To obtain sample 1/4 degree parameter data-sets for LIS' revision 4.0.2 using http:

1. From the "LIS 4.0" Section (See Section 6.2)

Follow the "Land Surface Parameters" link.

2. Get the “UMD Land/Sea Mask” file from the “Data-sets used by all LIS land surface models” section.
Use the “HTTP” link to save the UMD land/sea mask file into
\$WORKING/LIS/input/GVEG/1_4deg/
3. Get the “UMD Vegetation Classification map” file from the “Data-sets used by all LIS land surface models” section.
Use the “HTTP” link to save the UMD tile-space vegetation file into
\$WORKING/LIS/input/GVEG/1_4deg/
4. Get the “Soil color” file from the “Data-sets used by all LIS land surface models” section.
Use the “HTTP” link to save the soil color file into
\$WORKING/LIS/input/BCS/1_4deg/
5. Get the “Soil clay fraction” file from the “Data-sets used by all LIS land surface models” section.
Use the “HTTP” link to save the soil clay fraction file into
\$WORKING/LIS/input/BCS/1_4deg/
6. Get the “Soil sand fraction” file from the “Data-sets used by all LIS land surface models” section.
Use the “HTTP” link to save the soil sand fraction file into
\$WORKING/LIS/input/BCS/1_4deg/
7. Get the “Canopy height look-up table” from the “CLM Data-sets” section.
Use the html link to save the canopy height look-up table as *clm2-ptcanhts.txt* into
\$WORKING/LIS/input/BCS/clm_parms
8. Get the “Vegetation classification look-up table” from the “CLM Data-sets” section.
Use the html link to save the vegetation classification look-up table as *umdvegparam.txt* into
\$WORKING/LIS/input/BCS/clm_parms
9. Get the “Monthly Leaf Area Index” files from the “CLM Data-sets” section.
Follow the html link.
Then save the 12 monthly leaf area index files into *\$WORKING/LIS/input/AVHRR_LAI*
Return to main parameter data page.
10. Get the “Monthly Stem Area Index” files from the “CLM Data-sets” section.
Follow the html link. (The same link as for LAI.)

Then save the 12 monthly stem area index files into *\$WORKING/LIS/input/AVHRR_LAI*
Return to main parameter data page.

11. Get the “Vegetation look-up table” from the “Noah Data-sets” section.
Use the html link to save the vegetation look-up table as *noah.vegparms.txt* into
\$WORKING/LIS/input/BCS/noah_parms
12. Get the “Soil look-up table” from the “Noah Data-sets” section.
Use the html link to save the soil look-up table as *noah.soilparms.txt* into
\$WORKING/LIS/input/BCS/noah_parms
13. Get the “Quarterly albedo climatology” from the “Noah Data-sets” section.
Follow the html link.
Then save the 4 quarterly albedo climatology files into
\$WORKING/LIS/input/BCS/1_4deg/NOAH
Return to main parameter data page.
14. Get the “Monthly greenness fraction climatology” from the “Noah Data-sets” section.
Follow the html link.
Then save the 12 monthly greenness fraction climatology files into
\$WORKING/LIS/input/BCS/1_4deg/NOAH
Return to main parameter data page.
15. Get the “Maximum snow albedo” from the “Noah Data-sets” section.
Use the “HTTP” link to save the maximum snow albedo files into
\$WORKING/LIS/input/BCS/1_4deg/NOAH
16. Get the “Bottom temperature without elevation correction” files from the “Noah Data-sets” section.
Use the “HTTP” link to save the bottom temperature without elevation correction file into
\$WORKING/LIS/input/BCS/1_4deg/NOAH
17. Get the “Vegetation look-up table” from the “VIC Data-sets” section.
Use the html link to save the vegetation look-up table as *veg.lib.txt* into
\$WORKING/LIS/input/BCS/vic_parms
18. Get the “Soil look-up table” from the “VIC Data-sets” section.
Use the html link to save the soil look-up table as *soil.txt* into
\$WORKING/LIS/input/BCS/vic_parms

Note, files with names ending in *.gz* have been compressed using GNU's gzip. They must be “unzipped” before using. Files with names ending in *.tar.gz* have been packaged using GNU's tar and compressed using GNU's gzip. They must be “unzipped” and “untarred” before using.

6.2.3 Example: Downloading Sample 1/4 Deg. Parameter Data-sets Via GDS

To obtain sample 1/4 degree parameter data-sets for LIS' revision 4.0.2 using GDS, consider the following example:

Getting Noah “Bottom Temperature”, *tbot*, data.

Click on the “Tbot” link in the “GDS” column. You will see screen-shot 1.

Then using a DODS enabled GrADS client (see Section 4.3), issue the following GrADS' commands at the GrADS' prompt:

```
ga-> sdfopen http:// -- registered users will see a valid url --
ga-> set fwex
ga-> set t 1
ga-> set x 1 1440
ga-> set y 1 600
ga-> set fwrite -be -sq tbot.bfsa
ga-> set gxout fwrite
ga-> d tbot
ga-> quit
```

where:

“http:// – registered users will see a valid url –” comes from the “DODS URL:” line.

“set t 1” comes from the “Time:” line – (1 points).

“set x 1 1440” comes from the “Longitude:” line – (1440 points, avg. res. 0.25°).

“set y 1 600” comes from the “Latitude:” line – (600 points, avg. res. 0.25°).

“d tbot” comes from the list of “Variables:”.

This will save the *tbot* data onto your local disk in the file *tbot.bfsa*.

The other data-sets may be obtained from LIS' public GDS in a similar manner. Note, not all the data-sets are in a format suitable for serving through a GDS. Those data-sets must be obtained using the steps outlined in the previous “http” example (Section 6.2.2).

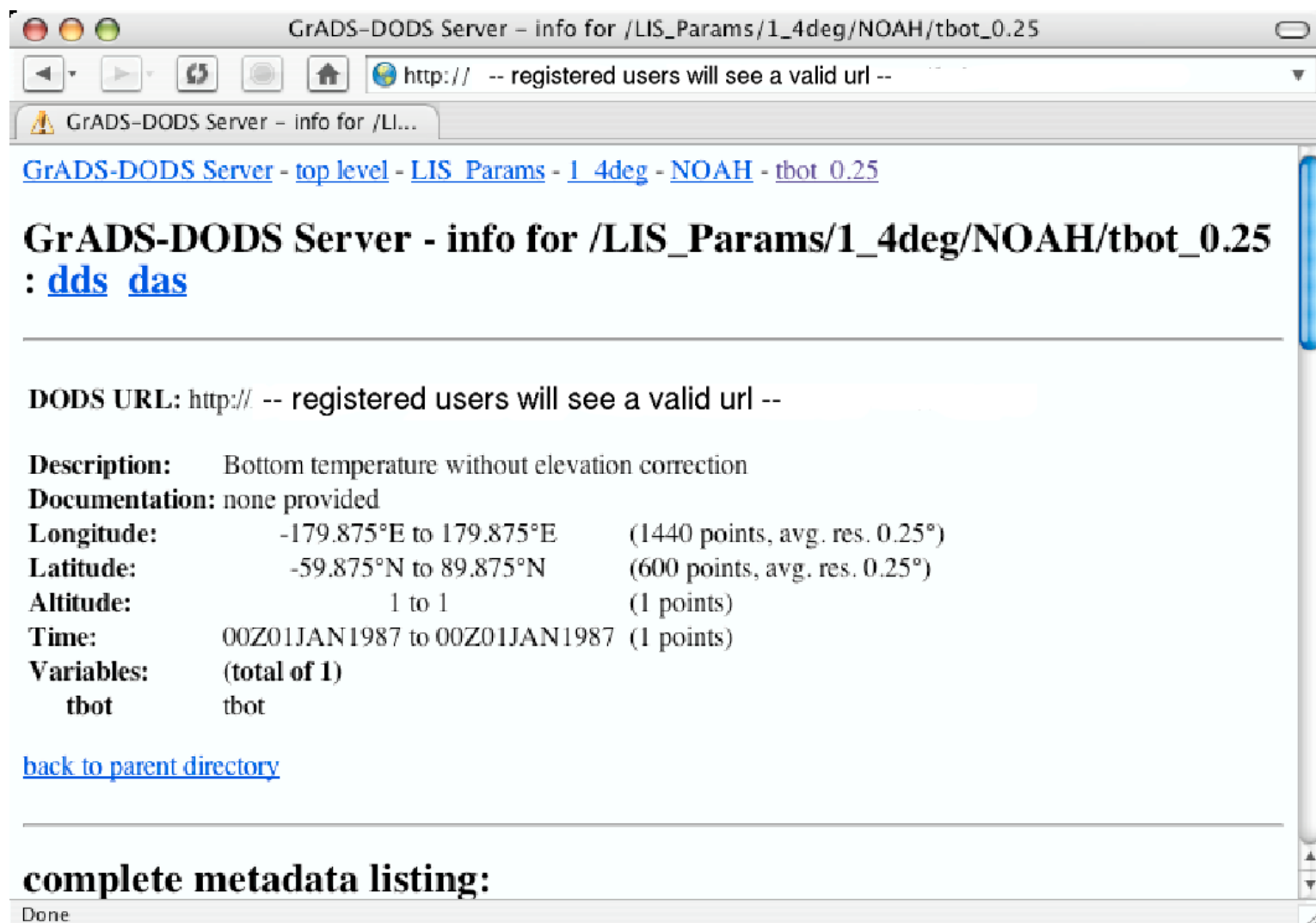


Figure 1: Screen-shot of tbot GDS link

6.2.4 Downloading Sample Forcing Data-sets

As mentioned earlier, the land surface models in LIS are forced by model-derived output and satellite and ground-based observations. Sample data-sets are available through the above link to atmospheric forcing data under the “LIS 4.0” data page.

6.2.5 Example: Downloading Sample 1/4 Deg. Forcing Data-sets Via HTTP

To obtain sample 1/4 degree forcing data-sets for LIS’ revision 4.0.2 using http:

1. From the “LIS 4.0” Section (See Section 6.2)
Follow the “Atmospheric Forcing Data” link.
2. Get the “GDAS” file.
Use the “HTTP” link to save the GDAS forcing data-set into
\$WORKING/LIS/input/FORCING/GDAS/
Note, this is 1-month sample of GDAS forcing data from 01 June 2001 to 30 June 2001.
3. Get the “GEOS3” file.
Use the “HTTP” link to save the GEOS3 forcing data-set into
\$WORKING/LIS/input/FORCING/GEOS/
Note, this is 1-month sample of GDAS forcing data from 01 June 2001 to 30 June 2001.
4. Get the “GDAS Elevation Correction” file.
Follow the html link.
Use the “HTTP” link to save the GDAS elevation correction data differences file into
\$WORKING/LIS/input/FORCING/
Return to main forcing data page.
5. Get the “GEOS3 Elevation Correction” file.
Follow the html link.
Use the “HTTP” link to save the GEOS3 elevation correction data differences file into
\$WORKING/LIS/input/FORCING/
Return to main forcing data page.
6. Get the “GEOS4 Elevation Correction” file.
Follow the html link.

Use the “HTTP” link to save the GEOS4 elevation correction data differences file into

\$WORKING/LIS/input/FORCING/

Return to main forcing data page.

7. Get the “AGRMET shortwave flux” file.

Use the “HTTP” link to save the AGRMET shortwave flux file into

\$WORKING/LIS/input/FORCING/AGRMET/

Note, this is 1-month sample of AGRMET forcing data from 01 June 2001 to 30 June 2001.

8. Get the “AGRMET longwave flux” file.

Use the “HTTP” link to save the AGRMET longwave flux file into

\$WORKING/LIS/input/FORCING/AGRMET/

Note, this is 1-month sample of AGRMET forcing data from 01 June 2001 to 30 June 2001.

9. Get the “CMAP Precipitaion” file.

Use the “HTTP” link to save the CMAP precipitation file into

\$WORKING/LIS/input/FORCING/CMAP/

Note, this is 1-month sample of AGRMET forcing data from 01 June 2001 to 30 June 2001.

Note, files with names ending in *.gz* have been compressed using GNU’s gzip. They must be “unzipped” before using. Files with names ending in *.tar.gz* have been packaged using GNU’s tar and compressed using GNU’s gzip. They must be “unzipped” and “untarred” before using.

Note, to obtain data outside of this 1-month sample, you must use the GDS.

Obtaining the forcing data-sets via the GDS is similar to the example given in Section 6.2.3.

6.3 Downloading Sample Output Data-sets

In addition to the above sample parameter and forcing data-sets, LIS provides sample output data-sets that may be downloaded and used for comparison. These data-sets are available through LIS’ public data server. You must use a DODS enabled GrADS client (see Section 4.3) to access the data.

6.3.1 Example: Downloading Sample 1/4 Deg. Output Data-sets Via GDS

To obtain the sample 1/4 degree output data-sets from LIS’ revision 4.0.2 using GDS, consider the following example:

Getting the Noah GEOS output data.

1. Go to LIS' public data server
Go to `http://` – registered users will see a valid url –
2. Then follow the “LIS-DATA” → “Output” → “global” → “1_4deg” links.
3. Then click on the “NOAH-GEOS“ “info” link.

You will be presented with a page similar to screen-shot 1.

To get only the “net shortwave radiation” variable, issue the following GrADS' commands at the GrADS' prompt:

```
ga-> sdfopen http:// -- registered users will see a valid url --
ga-> set fwex
ga-> set t 1 8
ga-> set x 1 1440
ga-> set y 1 600
ga-> set fwrite -be -sq swnet.bfsa
ga-> set gxout fwrite
ga-> d swnet
ga-> quit
```

Again, where: “`http://` – registered users will see a valid url –” comes from the “DODS URL:” line.

To get all variables, issue the following GrADS' commands at the GrADS' prompt:

```
ga-> sdfopen http:// -- registered users will see a valid url --
ga-> set fwex
ga-> set t 1 8
ga-> set x 1 1440
ga-> set y 1 600
ga-> set fwrite -be -sq noah.geos.bfsa
ga-> set gxout fwrite
ga-> d swnet
ga-> d lwnet
ga-> d qle
ga-> d qh
ga-> d qg
ga-> d snowf
ga-> d rainf
ga-> d evap
ga-> d qs
ga-> d qsb
ga-> d qsm
ga-> d delsoilmoist
ga-> d delswc
ga-> d avgsurft
```



```

ga-> d albedo
ga-> d swe
ga-> d soilmoist1
ga-> d soilmoist2
ga-> d soilmoist3
ga-> d soilmoist4
ga-> d soilwet
ga-> d tveg
ga-> d esoil
ga-> d rootmoist
ga-> d wind
ga-> d rainfforc
ga-> d snowfforc
ga-> d tair
ga-> d qair
ga-> d psurf
ga-> d swdown
ga-> d lwdown
ga-> quit

```

Again, where: “http:// – registered users will see a valid url –” comes from the “DODS URL:” line.

See Section 6.2.3 for more details.

Downloading the “CLM-GEOS” and “VIC-GEOS” data-sets is similar.

6.3.2 Viewing The Sample 1/4 Deg. Output Data-sets

Instead of downloading these sample output data-sets, you may view the data on-line. Follow the instructions in Section 6.3.1 that take you to the “NOAH-GEOS” “info” page on LIS’ public data server. Then follow the “Visualize!” link for whichever variable you wish to see. You will be presented with an easy-to-use interface for examining a data-plot of your chosen variable.

7 Building the Executable

This section describes how to build the source code and create LIS' executable – named LIS.

1. Perform the steps described in Section 5.
2. Compile the new GRIB library, *libw3.a*. You must edit the *Makefile* located in *\$WORKING/LIS/src/w3lib*. Uncomment the appropriate block of compiler flags, then run **gmake**.

3. Compile the LIS source code.

If you are building on an SGI or IBM system, simply run the compiling script.

- (a) Change directory into *\$WORKING/LIS/*.
% cd \$WORKING/LIS/
- (b) Run the compiling script.
% ./comp.csh

If you are building on a Linux PC system, you must edit the *Makefile* located in *\$WORKING/LIS/src/make*.

You must edit the definition of **ESMF_ARCH**. Currently it is defined to be **linux_absoft**, which specifies that you are using Absoft's Fortran compiler. If you are using Lahey's Fortran compiler, change **ESMF_ARCH** to **linux_lf95**.

Then run the compiling script.

- (a) Change directory into *\$WORKING/LIS/*.
% cd \$WORKING/LIS/
- (b) Run the compiling script.
% ./comp.csh

See Appendix B to see the Makefile.

7.1 Compiling In GSWP-2 Support

Note, GSWP-2 support in this release of LIS (4.0.2) should be considered beta.

In order for LIS to perform GSWP-2 experiments, you must compile in support for NetCDF reading and writing.

- You must edit the *misc.h* file located in *\$WORKING/LIS/src/make*.
Change the line that reads “#undef NETCDF” to “#define NETCDF”.
- You must edit the *Makefile* located in *\$WORKING/LIS/src/make*.
You must add **-L\$(LIB_NETCDF) -lnetcdf** to the **LD_FLAGS** variable.

Also, if you wish to use the CLM LSM, then change the line that reads “#undef CLM_GSWP_SUPPORT” to “#define CLM_GSWP_SUPPORT”.

7.2 Compiling In MPI Support

- Linux PC

If you are building on a Linux PC system, you must edit the *Makefile* located in *\$WORKING/LIS/src/make*.

You must change the definition of `MPI_PREFIX` to the directory where you installed MPICH. Currently `MPI_PREFIX` is set to */data1/jim/local/mpich-1.2.4-absoft*. Then you must uncomment the appropriate `FC` and `CC` definitions to use `mpif90` and `mpicc`.

- All Systems (including Linux PC)

- You must edit the *Makefile* located in *\$WORKING/LIS/src/make*. You must add either `-lmpich` or `-lmpi` (whichever is appropriate for your system) to the `LDFLAGS` variable.
- You must edit the *misc.h* file located in *\$WORKING/LIS/src/make*. Change the line that reads “`#undef SPMD`” to “`#define SPMD`”.

Then run the compiling script, described above.

7.3 General Build Instructions

This section describes how to build the LIS code on a platform other than those discussed in Section 3.

7.3.1 Required Software Libraries

In order to build the LIS executable, the following libraries must be installed on your system:

- Message Passing Interface (MPI) – If you wish to run the MPI-based running mode
 - vendor supplied, or
 - MPICH
(<http://http://www-unix.mcs.anl.gov/mpi/mpich/>)
- w3lib – Source include in the *source.tar.gz* file.

To install the MPI libraries, follow the instructions provided at the MPI URL listed above.

Note: Due to the mix of programming languages (Fortran and C) used by LIS, you may run into linking errors when building the LIS executable.

When compiling code using Absoft’s Pro Fortran SDK, set the following compiler options:

`-YEXT_NAMES=LCS -s -B108 -YCFRL=1`

-YEXT_NAMES=LCS sets the case of external names such as function and subroutine names to be lower case.

-s allocates local variables statically.

-B108 appends an underscore to function and subroutine names.

-YCFRL=1 passes character arguments in a manner compatible with g77.

These must be set for each of the above libraries.

7.3.2 Notes on compiling MPICH

The following bash shell commands are used to compile the MPICH library on a Linux cluster. Note that you may have to tweak these commands depending on the version of MPICH you are compiling and on the version of compiler that you are using. Refer to the on-line instructions found on the MPICH web-site.

1. Absoft

```
% export FC=f90
% export F90=f90
% export FFLAGS="-YEXT_NAMES=LCS -s -B108 -YCFRL=1"
% export FFLAGS90="-YEXT_NAMES=LCS -s -B108 -YCFRL=1"
% export RSHCOMMAND=ssh
% ./configure --with-arch=LINUX --with-device=ch_p4 --prefix=/some/shared/directory
% make
% make install
```

2. Lahey % export FC=lf95

```
% export F90=lf95
% export RSHCOMMAND=ssh
% ./configure --with-arch=LINUX --with-device=ch_p4 --prefix=/some/shared/directory
% make
% make install
```

3. Intel Fortran Compiler (version 8) % export FC=/opt/intel_fc_80/bin/ifort

```
% export F77=/opt/intel_fc_80/bin/ifort
% export F90=/opt/intel_fc_80/bin/ifort
% export FFLAGS="-nomixed_str_len_arg -names lowercase"
% export F90FLAGS="-nomixed_str_len_arg -names lowercase"
% export RSHCOMMAND=ssh
% ./configure --with-arch=LINUX --with-device=ch_p4 --prefix=/some/shared/directory
% make
% make install
```

Note that --prefix must be a directory that all compute nodes in your cluster can see.

7.3.3 Modifying the Makefile

This section lists the variables in the *Makefile* that must be set by the user before compiling.

Variable	Description
UNAMES	set by call to <code>uname</code> to determine what type of system you are using. Determine which variable (UNAMES or UMACHINE) will contain the needed information and use it.
UMACHINE	set by call to <code>uname</code> to determine what type of system you are using. Determine which variable (UNAMES or UMACHINE) will contain the needed information and use it.
MPI_PREFIX	path to where mpi libraries are installed
LIB_MPI	path to mpi libraries
INC_MPI	path to mpi header files
ESMF_DIR	path to where esmf libraries are installed
LIB_ESMF	path to esmf libraries
MOD_ESMF	path to esmf modules
ESMF_ARCH	system on which esmf libraries were compiled
FC	fortran compiler
CPP	C preprocessor
LIB_DIR	path to where lis libraries are installed
CPPFLAGS	flags for C preprocessor
CFLAGS	flags for C compiler
FFLAGS	flags for Fortran compiler
FOPTS	additional options for compiler and linker
LDFLAGS	flags for linker
NEW_ARCH_HERE	replace this with the type of system you are using, either the result from <code>uname -s</code> or <code>uname -m</code> . E.g., IRIX64 or i686

Note: For Linux architectures, the default `ESMF_ARCH` is set to be `linux_absoft`. For Linux systems using the Lahey Fortran compiler, `ESMF_ARCH` must be changed to `linux_lf95`.

7.4 Compiling GrADS-DODS Support

The above building instructions generate a LIS executable that reads input data off of local disk (the MPI-based running mode). To compile an executable that uses a GrADS-DODS server ¹ to retrieve input data files (the GDS-based running mode), you must edit the *Makefile*. Find the appropriate `FFLAGS` definition in the *Makefile*. Add `-DOPENDAP` to the end of the definition. Then follow the above building instructions.

¹This LIS distribution is configured to retrieve data using LIS' GrADS-DODS server.

7.5 Generating documentation

LIS code uses the ProTex documenting system [2]. The documentation in L^AT_EX format can be produced by typing `gmake doc` in the *\$WORKING/LIS/src/make* directory. This command produces documentation, generating all the files in *\$WORKING/LIS/doc* directory. These files can be easily converted to pdf or html formats using utilites such as `pdflatex` or `latex2html`.

8 Running The Executable

This section describes how to run the LIS executable. Once the LIS executable is built, a simulation can be performed using the *lis.crd* file. As described in Section 4, LIS is designed to support several modes of operation, in particular, single-process or parallel modes of operation.

The single-process version of LIS is executed by the following command issued in the *\$WORKING/LIS* directory.

```
% ./LIS
```

The parallel version of LIS must be run through an *mpirun* script. Assuming that MPI is installed correctly, the LIS simulation is carried out by the following command issued from in the *\$WORKING/LIS* directory.

```
% mpirun -np N ./LIS
```

The *-np N* flag indicates the number of processes to use in the run, where you replace *N* with the number of processes to use. On a multiprocessor machine, the parallel processing capabilities of LIS can be exploited using this flag.

See Section 8.3 for instructions on running LIS over the global 1 km domain.

8.1 Configuring Run Via LIS Card File

This section describes how to configure your LIS run by specifying the options in the *lis.crd* “card file”, which is nothing more than a set of Fortran namelists.

See Appendix A to see a sample lis card file.

8.1.1 driver namelist

The **driver** namelist of the card file consists of the following options:

```
LIS%d%DOMAIN
LIS%d%LSM
LIS%f%FORCE
LIS%d%LANDCOVER
LIS%d%SOIL
LIS%d%ELEV
LIS%p%LAI
```

LIS%d%DOMAIN specifies the projection of the grid for the run. Acceptable values are:

Value	Description
1	Lat/Lon projection – UMD AVHRR order
2	Lat/Lon projection – GSWP-2 order

LIS%d%LSM specifies the land surface model to run. Acceptable values are:

Value	Description
1	Noah
2	CLM
3	VIC
4	mosaic
5	hyssib
6	ssib

LIS%f%FORCE specifies the forcing data source for the run. Acceptable values are:

Value	Description
1	GDAS
2	GEOS
3	ECMWF
4	NLDAS
5	GSWP
6	BERG

LIS%d%LANDCOVER specifies the vegetation classification data for the run. Acceptable values are:

Value	Description
1	UMD AVHRR
2	GSWP-2

LIS%d%S0IL specifies the soil classification data for the run. Acceptable values are:

Value	Description
2	FAO
3	STATSGO
4	GSWP-2

LIS%p%ELEV specifies elevation correction data source for the run. Acceptable values are:

Value	Description
1	gtopo30 based

LIS%p%LAI specifies the LAI data source for the run. Acceptable values are:

Value	Description
2	AVHRR-based LAI
4	GSWP-2 LAI

8.1.2 lis_run_inputs namelist

In the `lis_run_inputs` namelist of the card file these parameters may be reset:

LIS%o%EXPCODE
 LIS%p%VCLASS
 LIS%p%NT
 LIS%f%NF
 LIS%f%NMIF
 LIS%f%ECOR
 LIS%o%WFOR
 LIS%f%INTERP
 LIS%o%WSINGLE
 LIS%o%WPARAM
 LIS%o%WTIL
 LIS%o%WOUT
 LIS%o%STARTCODE
 LIS%t%SSS
 LIS%t%SMN
 LIS%t%SHR
 LIS%t%SDA
 LIS%t%SMO
 LIS%t%SYR
 LIS%t%ENDCODE
 LIS%t%ESS
 LIS%t%EMN
 LIS%t%EHR
 LIS%t%EDA
 LIS%t%EMO
 LIS%t%EYR
 LIS%t%TS
 LIS%d%UDEF
 LIS%o%ODIR
 LIS%o%DFILE
 LIS%o%start_yr
 LIS%f%GPCSRC
 LIS%f%RADSRC
 LIS%d%MAXT
 LIS%d%MINA

LIS%o%EXPCODE specifies the “experiment code number” for the run. It is used in constructing the name of the output directory for the run. Acceptable values are any 3 character string.

LIS%p%VCLASS specifies the type of vegetation classification used. The default value is 1 corresponding to the UMD classification.

LIS%p%NT specifies the number of vegetation types. Acceptable values are:

Value	Description
13	corresponding to the UMD vegetation type classification
16	corresponding to the IGBP vegetation type classification

`LIS%f%NF` specifies the number of forcing variables. LIS currently uses 10 variables to force the LSMs

`LIS%f%NMIF` specifies the number of forcing variables for model initialization. The default value is set to 15.

`LIS%o%ECOR` specifies whether to use elevation correction for forcing. Acceptable values are:

Value	Description
0	Do not use elevation correction for forcing
1	Use elevation correction for forcing

`LIS%o%WFOR` specifies whether to output the ALMA optional forcing variables. Acceptable values are:

Value	Description
0	Do not output forcing variables
1	Do output forcing variables

`LIS%f%INTERP` specifies the type of interpolation scheme to apply to the forcing data. Acceptable values are:

Value	Description
1	bilinear scheme
2	conservative scheme

`LIS%o%WSINGLE` specifies whether to write each variable to a separate file or bundle them together. Acceptable values are:

Value	Description
0	Write all output variables to a single file
1	Write each output variable in a separate file

Note that the “`LIS%o%WSINGLE = 1`” output mode was designed to handle some of our memory intensive experiments. If you wish to use this output mode for a regular run (not a 1 km global run), then you must set `LIS%o%WOUT = 1`. Also note that the value of `LIS%o%WTIL` is ignored. So if you set `LIS%o%WSINGLE = 1`, then your data will be written as a 1-D land vector.

`LIS%o%WPARAM` specifies whether to write output for parameters such as the dominant vegetation type, soil type, lai, albedo, gfrac, etc. Acceptable values are:

Value	Description
0	Do not write parameter output
1	Write parameter output

`LIS%o%WTIL` specifies whether to write output as a 1-D array containing only land points or as a 2-D array containing both land and water points. Acceptable values are:

Value	Description
0	Write output in a 2-D grid domain
1	Write output in a 1-D grid domain

LIS%o%WOUT specifies the output data format. Acceptable values are:

Value	Description
1	Write output in binary format
2	Write output in grib format
3	Write output in netcdf format
4	Write output in netcdf format following the GSWP-2 conventions

LIS%o%STARTCODE specifies if a restart mode is being used. Acceptable values are:

Value	Description
1	A restart mode is being used
2	A cold start mode is being used, no restart file read
3	A cold start mode is being used, no restart file read, use GSWP-2 initial soil temperatures

When the cold start option is specified, the program is initialized using the LSM-specific initial conditions (typically assumed uniform for all tiles). When a restart mode is used, it is assumed that a corresponding restart file is provided depending upon which LSM is used. The user also needs to make sure that the ending time of the simulation is greater than model time when the restart file was written. See Sections 8.1.17, 8.1.18, and 8.1.19 to see how to specify the restart file.

Parameters LIS%t%SSS, LIS%t%SMN, LIS%t%SHR, LIS%t%SDA, LIS%t%SMO, and LIS%t%SYR are used in constructing the starting time for the run. Acceptable values are:

Variable	Value	Description
LIS%t%SSS	integer 0 – 59	specifying starting second
LIS%t%SMN	integer 0 – 59	specifying starting minute
LIS%t%SHR	integer 0 – 23	specifying starting hour
LIS%t%SDA	integer 1 – 31	specifying starting day
LIS%t%SMO	integer 1 – 12	specifying starting month
LIS%t%SYR	integer 2001 – present	specifying starting year

LIS%o%ENDCODE specifies the termination condition for runs. Acceptable values are:

Value	Description
0	Terminate the program at real-time date (not currently available)
1	Terminate the program at the specified date

Parameters LIS%t%ESS, LIS%t%EMN, LIS%t%EHR, LIS%t%EDA, LIS%t%EMO, and LIS%t%EYR are used in constructing the ending time for the run. Acceptable values are:

Variable	Value	Description
LIS%t%ESS	integer 0 – 59	specifying ending second
LIS%t%EMN	integer 0 – 59	specifying ending minute
LIS%t%EHR	integer 0 – 23	specifying ending hour
LIS%t%EDA	integer 1 – 31	specifying ending day
LIS%t%EMO	integer 1 – 12	specifying ending month
LIS%t%EYR	integer 2001 – present	specifying ending year

LIS%t%TS specifies the time-step for the run. Acceptable values are:

Value	Description
900	15 minute time-step
1800	30 minute time-step
3600	60 minute time-step

LIS%o%UDEF specifies the undefined value. The default is set to -9999.

LIS%o%ODIR specifies the name of the top-level output directory. Acceptable values are any 40 character string. The default value of LIS%o%ODIR is set to OUTPUT. For simplicity, throughout the rest of this document, this top-level output directory shall be referred to by its default name, *\$WORKING/LIS/OUTPUT*.

LIS%o%DFILE specifies the name of run time diagnostic file. Acceptable values are any 40 character string.

LIS%f%GPCSRC specifies if an observed precipitation forcing scheme is used or not. Acceptable values are:

Value	Description
0	No observed precipitation scheme used
1	use NRL precipitation product
2	use HUFFMANN precipitation product
3	use PERSIAN precipitation product
4	use CMAP precipitation product
5	use CMORPH precipitation product

Currently only CMAP precipitation product is implemented in LIS. Other schemes are under development.

LIS%f%RADSRC specifies if an observed radiation forcing scheme is used or not. Acceptable values are:

Value	Description
0	No observed radiation scheme used
1	use AGRMET radiation product

Parameters LIS%d%MAXT and LIS%d%MINA define the subgrid variability. LIS%d%MAXT defines the maximum tiles per grid (this can be as many as 13, the number of land cover types in the UMD vegetation classification). In addition, users select the smallest percentage of a cell for which to create a tile. LIS%d%MINA defines

this parameter. The percentage value is expressed as a fraction.

8.1.3 `run_domain` namelist

The `run_domain` namelist of the card file specifies the running domain (domain over which the simulation is carried out). LIS uses an array called `run_dd` that contains these domain definition parameters.

Variable	Description
<code>run_dd(1)</code>	0 = Equidistant cylindrical
<code>run_dd(2)</code>	Latitude of the south-west grid-cell center for the running domain
<code>run_dd(3)</code>	Longitude of the south-west grid-cell center for the running domain
<code>run_dd(4)</code>	Latitude of the north-east grid-cell center for the running domain
<code>run_dd(5)</code>	Longitude of the north-east grid-cell center for the running domain
<code>run_dd(6)</code>	Latitudinal increment of the running domain
<code>run_dd(7)</code>	Longitudinal increment of the running domain

See Section 8.2 for an example of how to set these values.

8.1.4 `param_domain` namelist

Currently it is assumed that all the parameter data are defined on the same domain, and that this domain contains the running domain described above.

Note: Several types of parameter data now have their own grid definitions. These are described in following sections.

The `param_domain` namelist of the card file specifies the parameter domain. LIS uses an array called `param_dd` that contains these domain definition parameters.

Variable	Description
<code>param_dd(1)</code>	Latitude of the south-west grid-cell center for the parameter data domain
<code>param_dd(2)</code>	Longitude of the south-west grid-cell center for the parameter data domain
<code>param_dd(3)</code>	Latitude of the north-east grid-cell center for the parameter data domain
<code>param_dd(4)</code>	Longitude of the north-east grid-cell center for the parameter data domain
<code>param_dd(5)</code>	Latitudinal increment of the parameter data domain
<code>param_dd(6)</code>	Longitudinal increment of the parameter data domain

See Section 8.2 for an example of how to set these values.

8.1.5 landcover namelist

The **landcover** namelist of the card file specifies the land/sea mask and vegetation classification data. See Section 6 for instructions on how to obtain these parameter data-sets.

```
LIS%p%MFILE
LIS%p%VFILE
LIS%d%lc_gridDesc(1)
LIS%d%lc_gridDesc(2)
LIS%d%lc_gridDesc(3)
LIS%d%lc_gridDesc(4)
LIS%d%lc_gridDesc(5)
LIS%d%lc_gridDesc(6)
```

LIS%p%MFILE specifies the location of land/water mask file.

LIS%p%VFILE specifies the location of the vegetation classification file.

The LIS%d%lc_gridDesc defines the grid these data are on. Each element is defined similarly to its corresponding element in the **param_dd** array. See Section 8.1.4.

8.1.6 elevation namelist

The **elevation** namelist of the card file specifies the elevation correction data. See Section 6 for instructions on how to obtain these parameter data-sets.

```
LIS%p%ELEVFILE
LIS%d%elev_gridDesc(1)
LIS%d%elev_gridDesc(2)
LIS%d%elev_gridDesc(3)
LIS%d%elev_gridDesc(4)
LIS%d%elev_gridDesc(5)
LIS%d%elev_gridDesc(6)
```

LIS%p%ELEVFILE specifies the elevation difference between LIS and EDAS (Eta Data Assimilation System) model grids.

The LIS%d%elev_gridDesc defines the grid these data are on. Each element is defined similarly to its corresponding element in the **param_dd** array. See Section 8.1.4.

8.1.7 soils namelist

The **soils** namelist of the card file specifies the soil classification data. See Section 6 for instructions on how to obtain these parameter data-sets.

```
LIS%p%SAFILE
LIS%p%CLFILE
LIS%p%ISCFIL
```

LIS%p%P01FILE
 LIS%p%P02FILE
 LIS%p%P03FILE
 LIS%p%SIFILE
 LIS%p%SLFILE
 LIS%p%w_sat_file
 LIS%p%w_sat_matp_file
 LIS%p%w_sat_hydc_file
 LIS%p%w_bpower_file
 LIS%p%w_wilt_file
 LIS%p%soilclass_file
 lis%p%soilp_type
 LIS%p%soiltemp_init
 LIS%d%soil_gridDesc(1)
 LIS%d%soil_gridDesc(2)
 LIS%d%soil_gridDesc(3)
 LIS%d%soil_gridDesc(4)
 LIS%d%soil_gridDesc(5)
 LIS%d%soil_gridDesc(6)

LIS%p%SAFILE specifies the sand fraction map file.

LIS%p%CLFILE specifies the clay fraction map file.

LIS%p%ISCFIL specifies the soil color map file.

LIS%p%P01FILE specifies top-layer porosity.

LIS%p%P02FILE specifies mid-layer porosity.

LIS%p%P03FILE specifies bottom-layer porosity.

LIS%p%SIFILE specifies the silt map file.

LIS%p%SLFILE specifies the slope map file.

LIS%p%w_sat_file specifies the saturated water content file.

LIS%p%w_sat_matp_file specifies the saturated matric potential file.

LIS%p%w_sat_hydc_file specifies the saturated hydraulic conductivity file.

LIS%p%w_bpower_file specifies the B exponent file.

LIS%p%w_wilt_file specifies the wilting point file.

LIS%p%soilclass_file specifies the soil texture class file.

LIS%p%soiltemp_init specifies the GSWP-2 initial soil temperature file.

lis%p%soilp_type specifies the source of the soil parameters. Acceptable values are:

Value	Description
1	look-up table
2	GSWP-2 data files

The LIS%d%soil_gridDesc defines the grid these data are on. Each element is defined similarly to its corresponding element in the `param.dd` array. See Section 8.1.4.

8.1.8 lai namelist

The lai namelist of the card file specifies the Leaf/Stem Area Index data. See Section 6 for instructions on how to obtain these parameter data-sets.

```
LIS%p%AVHRRDIR  
LIS%p%MODISDIR  
LIS%p%gswplai
```

LIS%p%AVHRRDIR and LIS%p%MODISDIR specifies the source for reading in LAI/SAI data (real time monthly data or climatology) for AVHRR and MODIS data, respectively. Once the source directory is specified, the program looks for real time data. If the real time data is not available, climatology data is read in.

LIS%p%gswplai specifies the GSWP-2 LAI file.

8.1.9 gswp namelist

The gswp namelist of the card file specifies the GSWP-2 base forcing data. See Section 6 for instructions on how to obtain these forcing data-sets.

```
gswpdrv%tair  
gswpdrv%qair  
gswpdrv%wind  
gswpdrv%psurf  
gswpdrv%rainf  
gswpdrv%snowf  
gswpdrv%swdown  
gswpdrv%lwdown  
gswpdrv%albedo  
gswpdrv%gfrac
```

gswpdrv%tair specifies the GSWP-2 temperature forcing file.

gswpdrv%qair specifies the GSWP-2 specific humidity forcing file.

gswpdrv%wind specifies the GSWP-2 near surface wind speed forcing file.

gswpdrv%psurf specifies the GSWP-2 surface pressure forcing file.

gswpdrv%rainf specifies the GSWP-2 rainfall rate forcing file.

gswpdrv%snowf specifies the GSWP-2 snowfall rate forcing file.

gswpdrv%swdown specifies the GSWP-2 surface incident shortwave radiation forcing file.

gswpdrv%lwdown specifies the GSWP-2 surface incident longwave radiation forcing file.

gswpdrv%albedo specifies the GSWP-2 albedo parameter file.

gswpdrv%gfrac specifies the GSWP-2 greenness fraction parameter file.

8.1.10 geos namelist

The `geos` namelist of the card file specifies the GEOS base forcing data. See Section 6 for instructions on how to obtain these forcing data-sets.

```
geosdrv%GEOSDIR  
geosdrv%NROL  
geosdrv%NCOLD  
geosdrv%NMIF
```

`geosdrv%GEOSDIR` specifies the location of the GEOS forcing files.

`geosdrv%NCOLD` and `geosdrv%NROL` specify the native domain parameters of the GEOS forcing data. The map projection is specified in the driver modules defined for the GEOS routines.

`geosdrv%NMIF` specifies the number of forcing variables provided by GEOS at the model initialization step.

8.1.11 gdas namelist

The `gdas` namelist of the card file specifies the GDAS base forcing data. See Section 6 for instructions on how to obtain these forcing data-sets.

```
gdasdrv%GDASDIR  
gdasdrv%NROL  
gdasdrv%NCOLD  
gdasdrv%NMIF
```

`gdasdrv%GDASDIR` specifies the location of the GDAS forcing files.

`gdasdrv%NCOLD` and `gdasdrv%NROL` specify the native domain parameters of the GDAS forcing data. The map projection is specified in the driver modules defined for the GDAS routines.

`gdasdrv%NMIF` specifies the number of forcing variables provided by GDAS at the model initialization step.

8.1.12 nldas namelist

The `nldas` namelist of the card file specifies the NLDAS base forcing data.

```
nldasdrv%NLDASDIR  
nldasdrv%NROL  
nldasdrv%NCOLD
```

`nldasdrv%NLDASDIR` specifies the location of the NLDAS forcing files.

`nldasdrv%NCOLD` and `nldasdrv%NROL` specify the native domain parameters of the NLDAS forcing data. The map projection is specified in the driver modules defined for the NLDAS routines.

8.1.13 ecmwf namelist

The **ecmwf** namelist of the card file specifies the ECMWF base forcing data.

```
ecmwdrv%ECMWDIR  
ecmwdrv%NROL  
ecmwdrv%NCOLD  
ecmwdrv%NMIF
```

ecmwdrv%ECMWDIR specifies the location of the ECMWF forcing files.

ecmwdrv%NCOLD and **ecmwdrv%NROL** specify the native domain parameters of the ECMWF forcing data. The map projection is specified in the driver modules defined for the ECMWF routines.

ecmwdrv%NMIF specifies the number of forcing variables provided by ECMWF at the model initialization step.

8.1.14 berg namelist

The **berg** namelist of the card file specifies the ECMWF reanalysis base forcing data.

```
bergdrv%emaskfile  
bergdrv%nrol  
bergdrv%ncold
```

bergdrv%emaskfile specifies the location of the ECMWF reanalysis land/sea mask file.

bergdrv%ncold and **bergdrv%nrol** specify the native domain parameters of the ECMWF reanalysis forcing data. The map projection is specified in the driver modules defined for the berg routines.

8.1.15 cmap namelist

The **cmap** namelist of the card file specifies the CMAP precipitation data. See Section 6 for instructions on how to obtain these forcing data-sets.

```
cmapdrv%CMAPDIR  
cmapdrv%NROL  
cmapdrv%NCOLD
```

cmapdrv%CMAPDIR specifies the location of the CMAP forcing files.

cmapdrv%NCOLD and **cmapdrv%NROL** specifies the native domain parameters of the CMAP forcing data. The map projection is specified in the driver modules defined for the CMAP routines.

8.1.16 agrmet namelist

The **agrmet** namelist of the card file specifies the AGRMET radiation data. See Section 6 for instructions on how to obtain these forcing data-sets.

agrmetdrv%AGRMETDIR

agrmetdrv%AGRMETDIR specifies the location of the AGRMET forcing files.

8.1.17 clm2 namelist

The **clm2** namelist of the card file specifies the run-time options and data specific to the CLM2 land surface model. See Section 6 for instructions on how to obtain these parameter data-sets.

clmdrv%WRITEINTC2
clmdrv%CLM2_RFILE
clmdrv%CLM2_VFILE
clmdrv%CLM2_CHTFILE
clmdrv%CLM2_ISM
clmdrv%CLM2_IT
clmdrv%CLM2_ISCV

clmdrv%WRITEINTC2 defines the output interval for CLM. Acceptable values range from 1 to 24 hours. The typical value used in the LIS runs is 3 hours.

clmdrv%CLM2_RFILE specifies the CLM active restart file.

clmdrv%CLM2_VFILE specifies vegetation type parameters look-up table.

clmdrv%CLM2_CHTFILE specifies the canopy top and bottom heights (for each vegetation type) look-up table.

clmdrv%CLM2_ISM specifies the initial soil moisture wetness used in the cold start runs.

clmdrv%CLM2_ISCV specifies the initial snow mass used in the cold start runs.

8.1.18 noah namelist

The **noah** namelist of the card file specifies the run-time options and data specific to the Noah land surface model. See Section 6 for instructions on how to obtain these parameter data-sets.

noahdrv%WRITEINTN
noahdrv%NOAH_RFILE
noahdrv%NOAH_MGFILE
noahdrv%NOAH_ALBFILE
noahdrv%NOAH_VFILE
noahdrv%NOAH_SFILE
noahdrv%NOAH_MXSNAL
noahdrv%NOAH_TBOT
noahdrv%NOAH_ISM

```

noahdrv%NOAH_IT
noahdrv%NOAH_NVEGT
noahdrv%NOAH_NVEGP
noahdrv%NOAH_NSILP
noahdrv%albedo_type
noahdrv%gfrac_type
noahdrv%mxsnalb_type
noahdrv%tbot_type

```

noahdrv%WRITEINTN defines the output interval for Noah. Acceptable values range from 1 to 24 hours. The typical value used in the LIS runs is 3 hours.

noahdrv%NOAH_RFILE specifies the Noah active restart file.

In the noah namelist of the card file these parameters must correspond with the domain resolution set in the `param_domain` namelist:

```

noahdrv%NOAH_MGFILE
noahdrv%NOAH_ALBFILE

```

noahdrv%NOAH_MGFILE and noahdrv%NOAH_ALBFILE specify where to find Noah's monthly greenness fraction and quarterly albedo input parameter data files.

noahdrv%NOAH_VFILE specifies the Noah static vegetation parameter file.

noahdrv%NOAH_SFILE specifies the Noah soil parameter file.

noahdrv%NOAH_MXSNAL specifies the Noah max snow free albedo.

noahdrv%NOAH_TBOT specifies the Noah bottom temperature.

noahdrv%NOAH_ISM specifies the initial volumetric soil moisture used in the cold start runs.

noahdrv%NOAH_IT specifies the initial skin temperature used in the cold start runs.

noahdrv%NOAH_NVEGP specifies the number of vegetation types.

noahdrv%NOAH_NVEGP specifies the number of static vegetation parameters specified for each veg type.

noahdrv%NOAH_NSILP specifies the number of static soil parameters specified.

noahdrv%albedo_type specifies the type of albedo data. Acceptable values are:

Value	Description
1	NCEP quarterly albedo
2	GSWP-2 monthly albedo

noahdrv%gfrac_type specifies the type of greenness fraction data. Acceptable values are:

Value	Description
1	NCEP monthly greenness fraction
2	GSWP-2 monthly greenness fraction

`noahdrv%mxsnalb_type` specifies the type of maximum snow albedo data. Acceptable values are:

Value	Description
1	NCEP maximum snow albedo
2	GSWP-2 formatted maximum snow albedo

`noahdrv%tbot_type` specifies the type of bottom temperature data. Acceptable values are:

Value	Description
1	NCEP bottom temperature
2	GSWP-2 formatted bottom temperature

8.1.19 vic namelist

The `vic` namelist of the card file specifies the run-time options and data specific to the VIC land surface model. See Section 6 for instructions on how to obtain these parameter data-sets.

```

vicdrv%WRITEINTVIC
vicdrv%VIC_NLAYER
vicdrv%VIC_NNODE
vicdrv%VIC_SNOWBAND
vicdrv%VIC_ROOTZONES
vicdrv%vic_frozen_soil
vicdrv%VIC_SFILE
vicdrv%VIC_VEGLIBFILE
vicdrv%VIC_RFILE
vicdrv%VIC_DSMAPFILE
vicdrv%VIC_DSMAXMAPFILE
vicdrv%VIC_WSMAPFILE
vicdrv%VIC_INFILTMAPFILE
vicdrv%VIC_DEPTH1MAPFILE
vicdrv%VIC_DEPTH2MAPFILE
vicdrv%VIC_DEPTH3MAPFILE
vicdrv%vic_initial_surf_temp

```

`vicdrv%WRITEINTVIC` defines the output interval for VIC. Acceptable values range from 1 to 24 hours. The typical value used in LIS runs is 3 hours.

`vicdrv%VIC_NLAYER` specifies the number of soil layers in VIC.

`vicdrv%VIC_NNODE` specifies the number of soil thermal nodes in VIC.

`vicdrv%VIC_SNOWBAND` specifies the number of snow bands in VIC.

`vicdrv%VIC_ROOTZONES` specifies the number of root zones in VIC.

`vicdrv%vic_frozen_soil` boolean flag (0/1) that specifies whether or not to perform the “frozen soil” calculations.

`vicdrv%VIC_SFILE` specifies the VIC soil parameter file.

`vicdrv%VIC_VEGLIBFILE` specifies the VIC vegetation parameter file.
`vicdrv%VIC_RFILE` specifies the active restart file.
`vicdrv%VIC_DSMAFILE` specifies fraction of maximum subsurface flow rate.
`vicdrv%VIC_DSMAFILE` specifies maximum subsurface flow rate.
`vicdrv%VIC_WSMAPFILE` specifies fraction of maximum soil moisture.
`vicdrv%VIC_INFILTMAPFILE` specifies infiltration.
`vicdrv%VIC_DEPTH1MAPFILE` specifies layer 1 soil depth.
`vicdrv%VIC_DEPTH2MAPFILE` specifies layer 2 soil depth.
`vicdrv%VIC_DEPTH3MAPFILE` specifies layer 3 soil depth.
`vicdrv%vic_initial_surf_temp` specifies the initial surface temperature in cold start runs.

8.1.20 opendap namelist

`opendap_data_prefix` specifies the top level directory where the GDS data retrieving script, *getdata.pl*, will place the data-sets it transfers.

In addition to modifying the *lis.crd* file by hand, you may use LIS' card file generator to write a cardfile for you. To use the card file generator

1. Go to LIS' "Public Release Home Page"
 Go to <http://lis.gsfc.nasa.gov/>
 Follow the "Source Codes" link.
 Follow the "LIS 4.0.2 Code Release" link.
2. From LIS' "Public Release Home Page"
 Follow the "LIS card file generator" link.
3. Fill in the blanks
4. Use your browser's "save as" feature to save a copy of the card file
5. Copy the card file to `$WORKING/LIS/lis.crd`

8.2 Domain Example

This section describes how to compute the values for the `run_dd` and `param_dd` arrays.

First, we shall generate the values for the parameter data domain. These are the values for the `param_dd` array. LIS' parameter data is defined on a Latitude/Longitude grid, from -180 to 180 degrees longitude and from -60 to 90 degrees latitude.

For this example, consider running at $1/4$ deg resolution. The coordinates of the south-west and the north-east points are specified at the grid-cells' centers. Here the south-west grid-cell is given by the box $(-180, -60), (-179.750, -59.750)$. The center of this box is $(-179.875, -59.875)$.²

²Note, these coordinates are ordered (longitude, latitude).

```
param_dd(1)  = -59.875
param_dd(2)  = -179.875
```

The north-east grid-cell is given by the box (179.750, 89.750), (180, 90). Its center is (179.875, 89.875).

```
param_dd(3)  = 89.875
param_dd(4)  = 179.875
```

Setting the resolution (0.25 deg) gives

```
param_dd(5)  = 0.25
param_dd(6)  = 0.25
```

And this completely defines the parameter data domain.

Next, we shall generate the values for the running domain. These are the values for the `run_dd` array.

Since the parameter data is on a Latitude/Longitude grid, we set

```
run_dd(1)    = 0
```

If you wish to run over the whole domain defined by the parameter data domain then you simply set the values of `dd(2) – dd(7)` equal to the values given by `dd(8) – dd(13)`. This gives

```
run_dd(2)    = -59.875
run_dd(3)    = -179.875
run_dd(4)    = 89.875
run_dd(5)    = 179.875
run_dd(6)    = 0.25
run_dd(7)    = 0.25
```

Now say you wish to run only over the region given by (−97.6, 27.9), (−92.9, 31.9). Since the running domain is a sub-set of the parameter domain, it is also a Latitude/Longitude domain at 1/4 deg. resolution. Thus,

```
run_dd(6)    = 0.25
run_dd(7)    = 0.25
```

Now, since the running domain must fit onto the parameter domain, the desired running region must be expanded from (−97.6, 27.9), (−92.9, 31.9) to (−97.75, 27.75), (−92.75, 32.0). The south-west grid-cell for the running domain is the box (−97.75, 27.75), (−97.5, 28.0). Its center is (−97.625, 27.875); giving

```
run_dd(2)    = 27.875
run_dd(3)    = -97.625
```

The north-east grid-cell for the running domain is the box (−93, 31.75), (−92.75, 32.0). Its center is (−92.875, 31.875); giving

```
run_dd(4)    = 31.875
run_dd(5)    = -92.875
```

This completely defines the running domain.

Note, the LIS project has defined 5 km resolution to be 0.05 deg. and 1 km resolution to be 0.01 deg. If you wish to run at 5 km or 1 km resolution, redo the above example to compute the appropriate grid-cell values.

See Figure 2 for an illustration of adjusting the running grid. See Figures 3 and 4 for an illustration of the south-west and north-east grid-cells.

8.3 Running Over The 1 km Domain

Note, these instructions are specific to LIS' Linux cluster.

To run LIS over the global 1 km domain, you must use the special “1 km” scripts.

Before running the scripts you must edit the *farmer.conf* configuration file found in */data1/pool/control*. The variables to set are:

```
cardfile
lsm
exp
```

cardfile specifies the full path to the *lis.crd* card file.

lsm specifies the land surface model being run. Acceptable values are:

```
NOAH
CLM2
VIC
```

exp is a label used to distinguish the simulation. It is used to construct a “snap-shot” directory. This “snap-shot” directory contains information that records which sub-domain patches each compute node processes. This information is used to assemble the output.

After editing the *farmer.conf*, you then run the *create-bones.pl* and *start-farmer-dogs.pl* Perl scripts, which are also in */data1/pool/control*.

```
% cd /data1/pool/control
% ./create-bones.pl
% ./start-farmer-dogs.pl
```

create-bones.pl constructs the “pool” of sub-domains to process. *start-farmer-dogs.pl* starts the simulation.

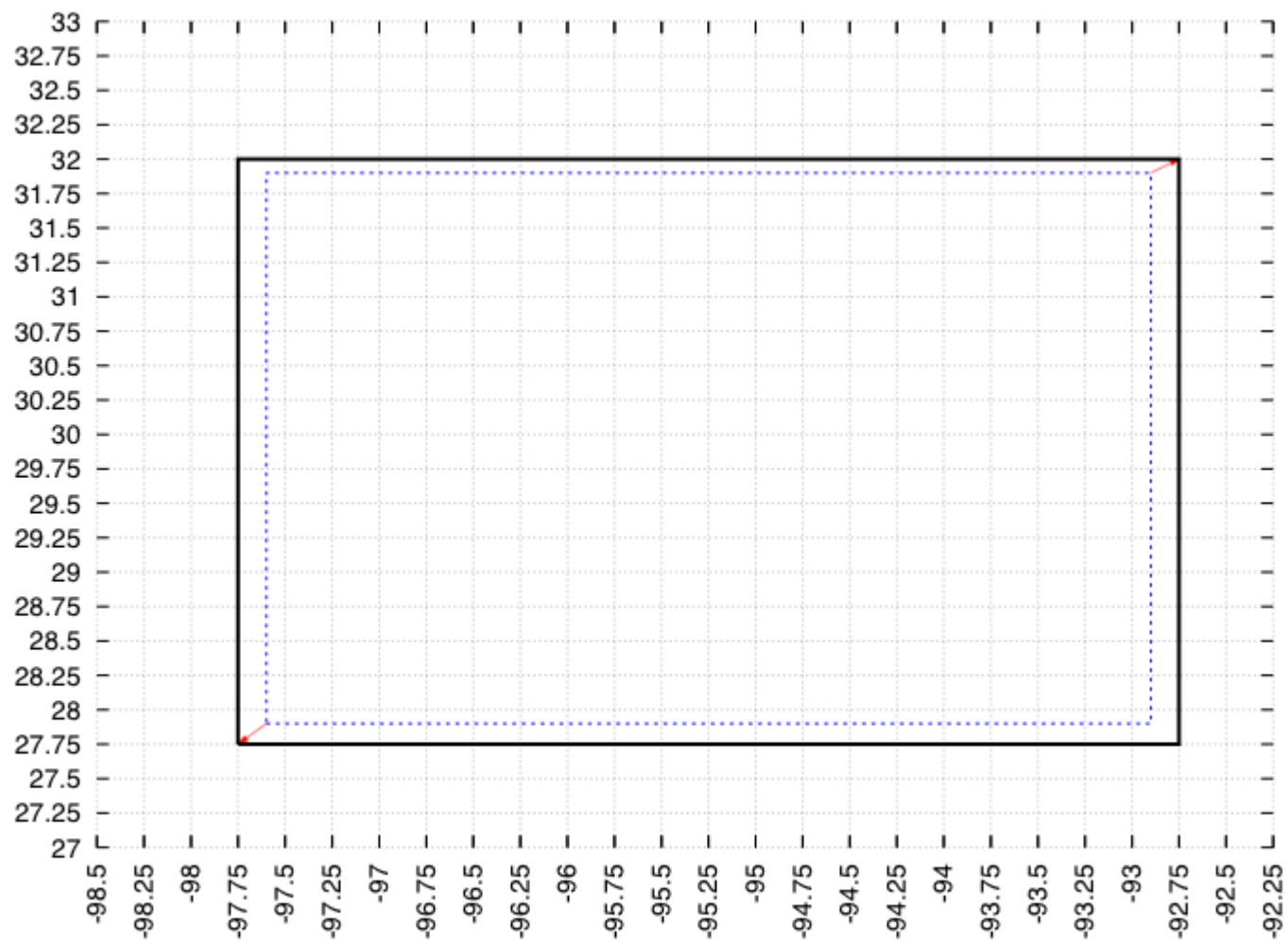


Figure 2: Illustration showing how to fit the desired running grid onto the actual grid

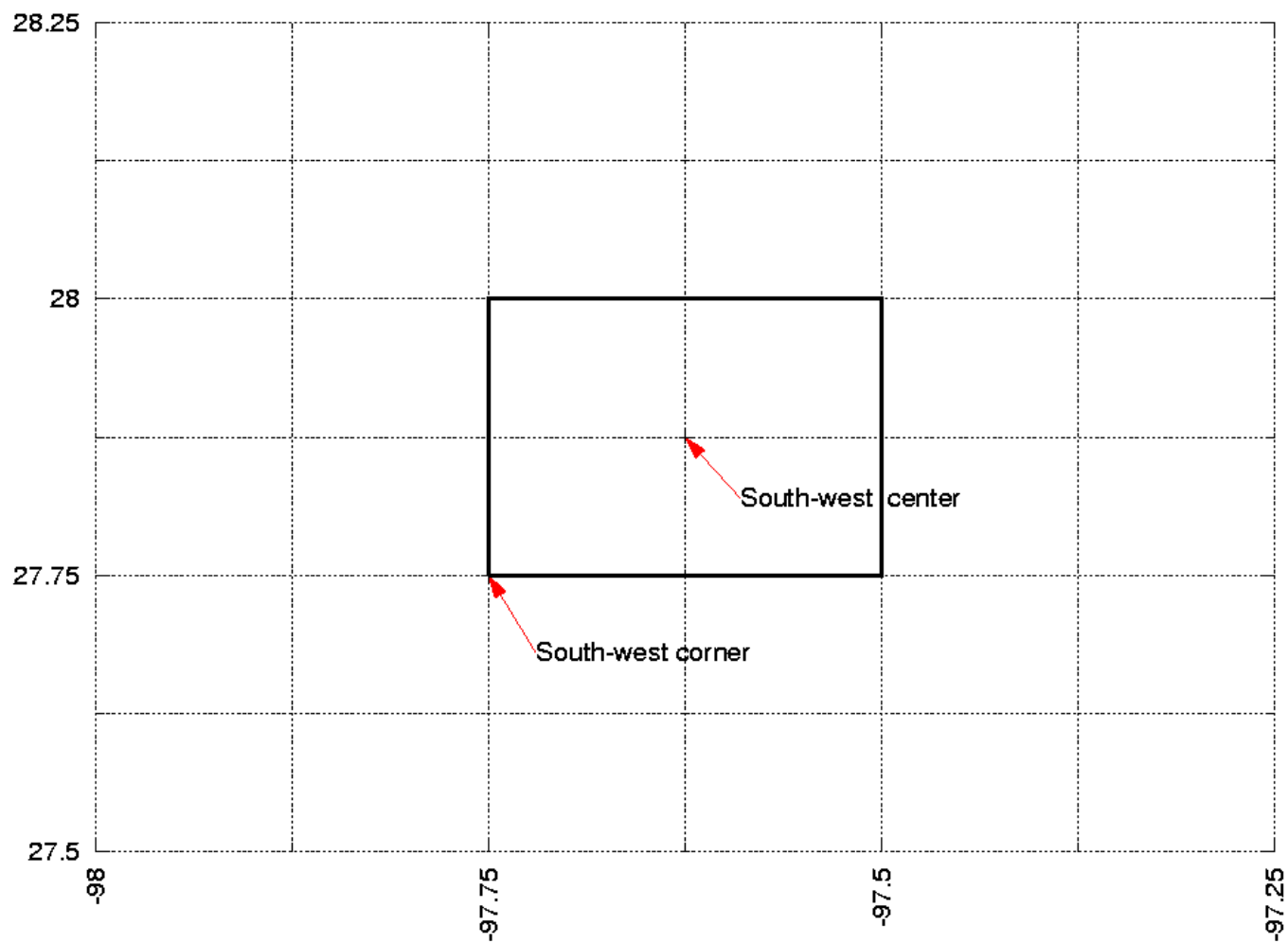


Figure 3: Illustration showing the south-west grid-cell corresponding to the example in Section 8.2

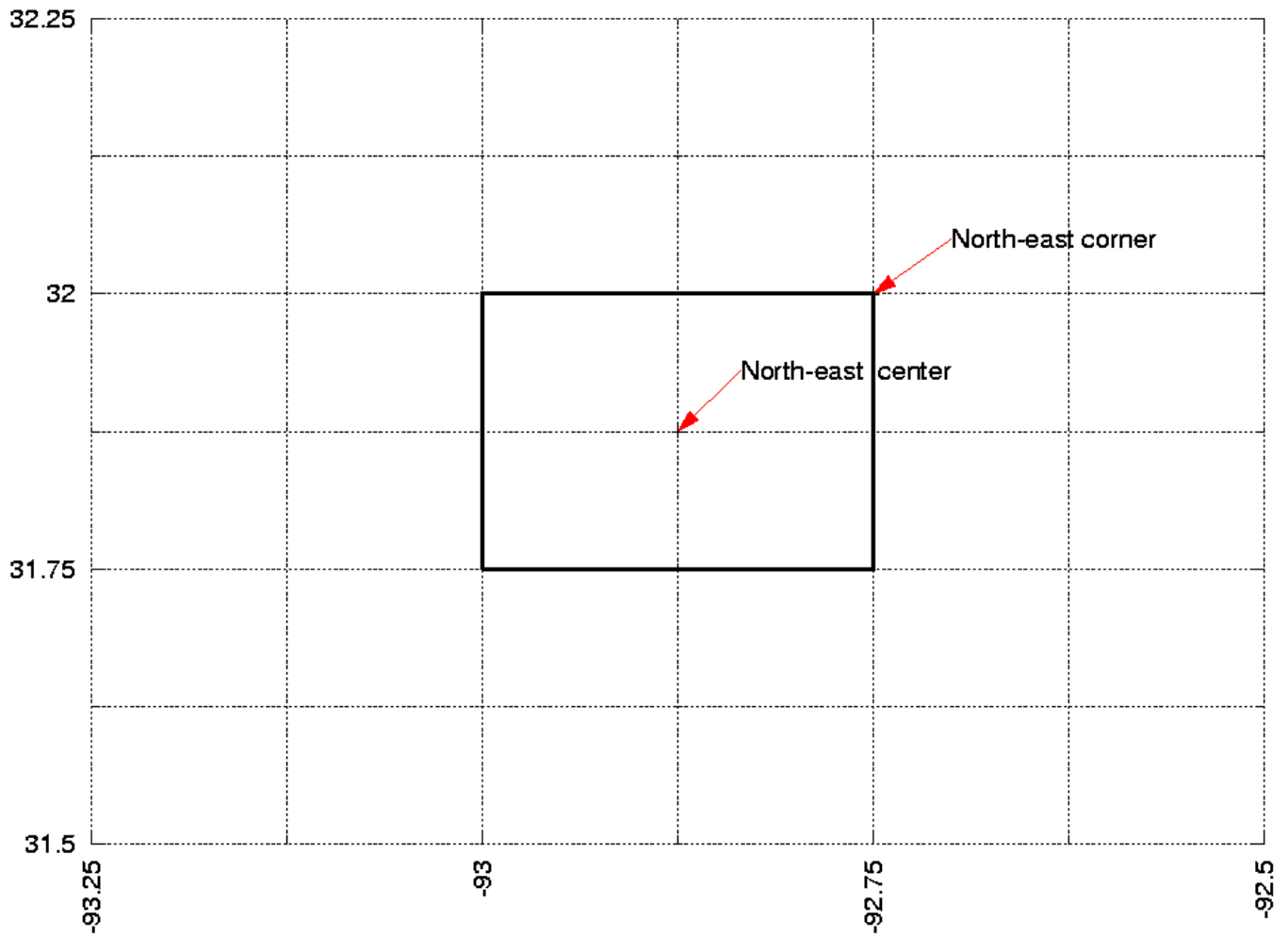


Figure 4: Illustration showing the north-east grid-cell corresponding to the example in Section 8.2

9 Output Data Processing

This section describes how to process the generated output.

The output data-sets created by running the LIS executable are written into sub-directories of the *\$WORKING/LIS/OUTPUT/* directory (created at run-time). These sub-directories are named *EXP999* (by default).

The output data consists of ASCII text files and model output in binary format.

For example, assume that you performed a “1/4 deg Noah with GEOS forcing” simulation for 11 June 2001, with an experiment code value of 999, and writing output as a 2-D array.

This run will produce a *\$WORKING/LIS/OUTPUT/EXP999/* directory. This directory will contain:

File Name	Synopsis
Noahstats.dat	Statistical summary of output
NOAH	Directory containing output data

The *NOAH* directory will contain sub-directories of the form *YYYY/YYYYMMDD*, where *YYYY* is a 4-digit year and *YYYYMMDD* is a date written as a 4-digit year, 2-digit month and a 2-digit day; both corresponding to the running dates of the simulation.

For this example, *NOAH* will contain a *2001/20010611* sub-directory.

Its contents are the output files generated by the executable. They are:

2001061100.gs4r

2001061103.gs4r

2001061106.gs4r

2001061109.gs4r

2001061112.gs4r

2001061115.gs4r

2001061118.gs4r

2001061121.gs4r

Note, each file-name contains a date-stamp marking the year, month, day, and hour that the data corresponds to. The output data files for CLM and VIC are similar.

The generated output can be written in a 2-D grid format or as a 1-d vector. See Section 8.1.2 for more details. If written as a 1-d vector, the output must be converted into a 2-d grid before it can be visualized. The *postproc.tar.gz*

contains 1-d to 2-d mapping files that GrADS can use for this purpose.³ See Section 5.4.

In this example, for 2-D output, simply copy the *noah.25.ctl* and *noah.25.gs* files into the *\$WORKING/LIS/OUTPUT/EXP999/NOAH/2001/20010611* directory. Then run GrADS:

```
% grads -blc "run noah.25.gs"
```

This will create data-plots of all the variables in *2001061121.gs4r*.

For 1-D output, you must edit the *noah.25.ctl* data descriptor file. Simply uncomment (remove the asterisk) the “PDEF” line. Then run GrADS as above.

³These mappings only work for “global” runs. If your simulation was over a sub-setted domain, you must either create your own pdef mapping file or simply write 2-d output.

9.1 CLM Output

This table lists the variables written by CLM. The output variables are written to conform to the standards specified by the Assistance for Land-surface Modelling activities (ALMA). See ALMA's web-site [3] for further details.

ALMA Mandatory Output

Number	Variable	Description	Units
1	SWnet	Net Shortwave Radiation	W/m ²
2	LWnet	Net Longwave Radiation	W/m ²
3	Qle	Latent Heat Flux	W/m ²
4	Qh	Sensible Heat Flux	W/m ²
5	Qg	Ground Heat Flux	W/m ²
6	Snowf	Snowfall rate	kg/m ² /s
7	Rainf	Rainfall rate	kg/m ² /s
8	Evap	Total Evapotranspiration	kg/m ² /s
9	Qs	Surface Runoff	kg/m ² /s
10	Qsb	Subsurface Runoff	kg/m ² /s
11	Qsm	Snowmelt	kg/m ² /s
12	DelSoilMoist	Change in soil moisture	kg/m ²
13	DelSWE	Change in snow water equivalent	kg/m ²
14	SnowT	Snow Temperature	K
15	VegT	Vegetation Canopy Temperature	K
16	BaresoilT	Temperature of bare soil	K
17	AvgSurfT	Average Surface Temperature	K
18	RadT	Surface Radiative Temperature	K
19	Albedo	Surface Albedo, All Wavelengths	-
20	SWE	Snow Water Equivalent	kg/m ²

Number	Variable	Description	Units
21	SoilTemp1	Average layer 1 soil temperature	K
22	SoilTemp2	Average layer 2 soil temperature	K
23	SoilTemp3	Average layer 3 soil temperature	K
24	SoilTemp4	Average layer 4 soil temperature	K
25	SoilTemp5	Average layer 5 soil temperature	K
26	SoilTemp6	Average layer 6 soil temperature	K
27	SoilTemp7	Average layer 7 soil temperature	K
28	SoilTemp8	Average layer 8 soil temperature	K
29	SoilTemp9	Average layer 9 soil temperature	K
30	SoilTemp10	Average layer 10 soil temperature	K
31	SoilMoist1	Average layer 1 soil moisture	kg/m ²
32	SoilMoist2	Average layer 2 soil moisture	kg/m ²
33	SoilMoist3	Average layer 3 soil moisture	kg/m ²
34	SoilMoist4	Average layer 4 soil moisture	kg/m ²
35	SoilMoist5	Average layer 5 soil moisture	kg/m ²
36	SoilMoist6	Average layer 6 soil moisture	kg/m ²
37	SoilMoist7	Average layer 7 soil moisture	kg/m ²
38	SoilMoist8	Average layer 8 soil moisture	kg/m ²
39	SoilMoist9	Average layer 9 soil moisture	kg/m ²
40	SoilMoist10	Average layer 10 soil moisture	kg/m ²
41	RootMoist	Root zone soil moisture	kg/m ²
42	SoilWet	Total Soil Wetness	-
43	TVeg	Vegetation transpiration	kg/m ² /s
44	ECanop	Interception evaporation	kg/m ² /s
45	ESoil	Bare soil evaporation	kg/m ² /s
46	Canopint	Total canopy water storage	kg/m ²
47	ACond	Aerodynamic conductance	m/s

ALMA Optional Forcing Output

Number	Variable	Description	Units
48	Wind	Near surface wind magnitude	m/s
49	Rainf	Rainfall rate	kg/m ² /s
50	Snowf	Snowfall rate	kg/m ² /s
51	Tair	Near surface air temperature	K
52	Qair	Near surface specific humidity	kg/kg
53	PSurf	Surface pressure	Pa
54	SWdown	Surface incident shortwave radiation	W/m ²
55	LWdown	Surface incident longwave radiation	W/m ²

9.2 Noah Output

This table lists the variables written by Noah. The output variables are written to conform to the standards specified by the Assistance for Land-surface Modelling activities (ALMA). See ALMA's web-site [3] for further details.

ALMA Mandatory Output

Number	Variable	Description	Units
1	SWnet	Net Shortwave Radiation	W/m ²
2	LWnet	Net Longwave Radiation	W/m ²
3	Qle	Latent Heat Flux	W/m ²
4	Qh	Sensible Heat Flux	W/m ²
5	Qg	Ground Heat Flux	W/m ²
6	Snowf	Snowfall rate	kg/m ² /s
7	Rainf	Rainfall rate	kg/m ² /s
8	Evap	Total Evapotranspiration	kg/m ² /s
9	Qs	Surface Runoff	kg/m ² /s
10	Qsb	Subsurface Runoff	kg/m ² /s
11	Qsm	Snowmelt	kg/m ² /s
12	DelSoilMoist	Change in soil moisture	kg/m ²
13	DelSWE	Change in snow water equivalent	kg/m ²
14	AvgSurfT	Average Surface Temperature	K
15	Albedo	Surface Albedo, All Wavelengths	-
16	SWE	Snow Water Equivalent	kg/m ²
17	SoilTemp1	Average layer 1 soil temperature	K
18	SoilTemp2	Average layer 2 soil temperature	K
19	SoilTemp3	Average layer 3 soil temperature	K
20	SoilTemp4	Average layer 4 soil temperature	K
21	SoilMoist1	Average layer 1 soil moisture	kg/m ²
22	SoilMoist2	Average layer 2 soil moisture	kg/m ²
23	SoilMoist3	Average layer 3 soil moisture	kg/m ²
24	SoilMoist4	Average layer 4 soil moisture	kg/m ²
25	SoilWet	Total Soil Wetness	-

26	ECanop	Interception evaporation	kg/m ² /s
27	TVeg	Vegetation transpiration	kg/m ² /s
28	ESoil	Bare soil evaporation	kg/m ² /s
29	RootMoist	Root zone soil moisture	kg/m ²
30	Canopint	Total canopy water storage	kg/m ²

ALMA Optional Forcing Output

Number	Variable	Description	Units
31	Wind	Near surface wind magnitude	m/s
32	Rainf	Rainfall rate	kg/m ² /s
33	Snowf	Snowfall rate	kg/m ² /s
34	Tair	Near surface air temperature	K
35	Qair	Near surface specific humidity	kg/kg
36	PSurf	Surface pressure	Pa
37	SWdown	Surface incident shortwave radiation	W/m ²
38	LWdown	Surface incident longwave radiation	W/m ²

9.3 VIC Output

This table lists the variables written by VIC. The output variables are written to conform to the standards specified by the Assistance for Land-surface Modelling activities (ALMA). See ALMA's web-site [3] for further details.

ALMA Mandatory Output

Number	Variable	Description	Units
1	SWnet	Net Shortwave Radiation	W/m ²
2	LWnet	Net Longwave Radiation	W/m ²
3	Qle	Latent Heat Flux	W/m ²
4	Qh	Sensible Heat Flux	W/m ²
5	Qg	Ground Heat Flux	W/m ²
6	Rainf	Rainfall rate	kg/m ² /s
7	Snowf	Snowfall rate	kg/m ² /s
8	Evap	Total Evapotranspiration	kg/m ² /s
9	Qs	Surface Runoff	kg/m ² /s
10	Qsb	Subsurface Runoff	kg/m ² /s
11	Qfz	Re-freezing of water in the snow	kg/m ² /s
12	SnowT	Snow Temperature	K
13	AvgSurfT	Average Surface Temperature	K
14	RadT	Surface Radiative Temperature	K
15	Albedo	Surface Albedo, All Wavelengths	-
16	SoilTemp1	Average layer 1 soil temperature	kg/m ²
17	SoilTemp2	Average layer 2 soil temperature	kg/m ²
18	SoilTemp3	Average layer 3 soil temperature	kg/m ²
19	SoilMoist1	Average layer 1 soil moisture	kg/m ²
20	SoilMoist2	Average layer 2 soil moisture	kg/m ²
21	SoilMoist3	Average layer 3 soil moisture	kg/m ²
22	TVeg	Vegetation transpiration	kg/m ² /s
23	ESoil	Bare soil evaporation	kg/m ² /s
24	SoilWet	Total Soil Wetness	-
25	RootMoist	Root zone soil moisture	kg/m ²

26	SWE	Snow Water Equivalent	kg/m ²
27	Qsm	Snowmelt	kg/m ² /s
28	DelSoilMoist	Change in soil moisture	kg/m ²
29	DelSWE	Change in snow water equivalent	kg/m ²
30	ACond	Aerodynamic conductance	m/s

ALMA Optional Forcing Output

Number	Variable	Description	Units
31	Wind	Near surface wind magnitude	m/s
32	Rainf	Rainfall rate	kg/m ² /s
33	Snowf	Snowfall rate	kg/m ² /s
34	Tair	Near surface air temperature	K
35	Qair	Near surface specific humidity	kg/kg
36	PSurf	Surface pressure	Pa
37	SWdown	Surface incident shortwave radiation	W/m ²
38	LWdown	Surface incident longwave radiation	W/m ²

A LIS Card File

This is a sample LIS card file. This card file configures LIS to run the Noah land surface model at 1/4 degree resolution using GEOS forcing data. The run starts at 21 hours 10 June 2001 and ends at 21 hours 11 June 2001. It will compute over the region given by the box $(-97.75, 27.75)$, $(-92.75, 32.0)$ degrees longitude/latitude.

```
&driver
LIS%d%DOMAIN      = 1
LIS%d%LSM         = 1
LIS%f%FORCE       = 2
LIS%d%LANDCOVER   = 1
LIS%d%SOIL        = 2
LIS%d%ELEV        = 1
LIS%p%LAI         = 2
/

&lis_run_inputs
LIS%o%EXPCODE      = "999"
LIS%p%VCLASS       = 1
LIS%p%NT           = 13
LIS%f%NF           = 10
LIS%f%NMIF         = 15
LIS%f%ECOR         = 1
LIS%o%WFOR         = 1
LIS%f%INTERP       = 1
LIS%o%WSINGLE       = 0
LIS%o%WPARAM       = 0
LIS%o%WTIL         = 0
LIS%o%WOUT         = 1
LIS%o%STARTCODE    = 2
LIS%t%SSS          = 0
LIS%t%SMN          = 00
LIS%t%SHR          = 21
LIS%t%SDA          = 10
LIS%t%SMO          = 06
LIS%t%SYR          = 2001
LIS%t%ENDCODE      = 1
LIS%t%ESS          = 0
LIS%t%EMN          = 00
LIS%t%EHR          = 21
LIS%t%EDA          = 11
LIS%t%EMO          = 06
LIS%t%EYR          = 2001
```

```

    LIS%t%TS           = 1800
    LIS%d%UDEF         = -9999.
    LIS%o%ODIR         = "/tmp/OUTPUT"
    LIS%o%DFILE        = "lisdiag"
    LIS%o%start_yr     = 2001
    LIS%f%GPCSRC       = 0
    LIS%f%RADSRC       = 0
    LIS%d%MAXT         = 1
    LIS%d%MINA         = 0.05
/

&run_domain
run_dd(1)  = 0
run_dd(2)  = -59.875
run_dd(3)  = -179.875
run_dd(4)  = 89.875
run_dd(5)  = 179.875
run_dd(6)  = 0.25
run_dd(7)  = 0.25
/

&param_domain
param_dd(1)  = -59.875
param_dd(2)  = -179.875
param_dd(3)  = 89.875
param_dd(4)  = 179.875
param_dd(5)  = 0.25
param_dd(6)  = 0.25
/

&landcover
LIS%p%MFILE      = "input/GVEG/1_4deg/UMD_AVHRR60mask0.25.bfsa"
LIS%p%VFILE      = "input/GVEG/1_4deg/UMD_AVHRR60G0.25.bfsa"
LIS%d%lc_gridDesc(1) = -59.875
LIS%d%lc_gridDesc(2) = -179.875
LIS%d%lc_gridDesc(3) = 89.875
LIS%d%lc_gridDesc(4) = 179.875
LIS%d%lc_gridDesc(5) = 0.25
LIS%d%lc_gridDesc(6) = 0.25
/

&elevation
LIS%p%ELEVFILE      = "input/GVEG/1_4deg/geos3_diff.1gd4r"
LIS%d%elev_gridDesc(1) = -59.875
LIS%d%elev_gridDesc(2) = -179.875
LIS%d%elev_gridDesc(3) = 89.875

```

```

LIS%d%elev_gridDesc(4) = 179.875
LIS%d%elev_gridDesc(5) = 0.25
LIS%d%elev_gridDesc(6) = 0.25
/

&soils
LIS%p%SAFILE      = "input/BCS/1_4deg/SOILS/sandfao.1gd4r"
LIS%p%CLFILE      = "input/BCS/1_4deg/SOILS/clayfao.1gd4r"
LIS%p%ISCFIL      = "input/BCS/1_4deg/SOILS/soicolfao.1gd4r"
LIS%p%P01FILE     = "input/BCS/1_4deg/SOILS/porfaot.1gd4r"
LIS%p%P02FILE     = "input/BCS/1_4deg/SOILS/porfaom.1gd4r"
LIS%p%P03FILE     = "input/BCS/1_4deg/SOILS/porfaob.1gd4r"
LIS%p%SIFILE      = "input/BCS/1_4deg/SOILS/siltfao.1gd4r"
LIS%p%SLFILE      = "input/BCS/1_4deg/ELEVATION/slope.1gd4r"
LIS%p%w_sat_file   = "gswp2data/Fixed/W_sat.nc"
LIS%p%w_sat_matp_file = "gswp2data/Fixed/W_sat_matp_CEA84.nc"
LIS%p%w_sat_hydc_file = "gswp2data/Fixed/W_sat_hydc.nc"
LIS%p%w_bpower_file = "gswp2data/Fixed/W_bpower_CEA84.nc"
LIS%p%w_wilt_file  = "gswp2data/Fixed/W_wilt.nc"
LIS%p%soilclass_file = "gswp2data/Fixed/SoilClass.nc"
lis%p%soilp_type   = 1
LIS%p%soiltemp_init = "gswp2data/Fixed/SoilTemp_init.nc"
LIS%d%soil_gridDesc(1) = -59.875
LIS%d%soil_gridDesc(2) = -179.875
LIS%d%soil_gridDesc(3) = 89.875
LIS%d%soil_gridDesc(4) = 179.875
LIS%d%soil_gridDesc(5) = 0.25
LIS%d%soil_gridDesc(6) = 0.25
/

&lai
LIS%p%AVHRRDIR    = "input/AVHRR_LAI/"
LIS%p%MODISDIR    = "input/MODIS_LAI/"
LIS%p%gswplai     = "gswp2data/LAI_uk/LAI_uk.nc"
/

&gswp
gswpdrv%tair      = "gswp2data/Tair_cru/Tair_cru"
gswpdrv%qair      = "gswp2data/Qair_cru/Qair_cru"
gswpdrv%wind      = "gswp2data/Wind_ncep/Wind_ncep"
gswpdrv%psurf     = "gswp2data/PSurf_ecor/PSurf_ecor"
gswpdrv%rainf     = "gswp2data/Rainf_gswp/Rainf_gswp"
gswpdrv%snowf     = "gswp2data/Snowf_gswp/Snowf_gswp"
gswpdrv%swdown    = "gswp2data/SWdown_srb/SWdown_srb"
gswpdrv%lwdown    = "gswp2data/LWdown_srb/LWdown_srb"
gswpdrv%albedo    = "gswp2data/Albedo_csu/Albedo_csu.nc"

```

```

gswpdrv%gfrac = "gswp2data/Grn_uk/grnFrac_uk.nc"
/

&geos
geosdrv%GEOSDIR = "input/FORCING/GEOS/BEST_LK"
geosdrv%NROL = 181
geosdrv%NCOLD = 360
geosdrv%NMIF = 13
/

&gdas
gdasdrv%GDASDIR = "input/FORCING/GDAS"
gdasdrv%NROL = 256
gdasdrv%NCOLD = 512
gdasdrv%NMIF = 15
/

&nldas
nldasdrv%NLDASDIR = "input/FORCING/NLDAS"
nldasdrv%NROL = 224
nldasdrv%NCOLD = 464
/

&ecmwf
ecmwfdrv%ECMWFDIR = "input/FORCING/ECMWF"
ecmwfdrv%NROL = 601
ecmwfdrv%NCOLD = 1440
ecmwfdrv%NMIF = 13
/

&berg
bergdrv%emaskfile = "input/BCS/ecmwf_land_sea.05"
bergdrv%nrol = 360
bergdrv%ncold = 720
/

&cmap
cmapdrv%CMAPDIR = "input/FORCING/CMAP"
cmapdrv%NROL = 256
cmapdrv%NCOLD = 512
/

&agrmet
agrmetdrv%AGRMETDIR = "input/FORCING/AGRMET"
/

```



```

&template
templatedrv%WRITEINT = 3
/

&clm2
clmdrv%WRITEINTC2 = 3
clmdrv%CLM2_RFILE = "clm2.rst"
clmdrv%CLM2_VFILE = "input/BCS/clm_parms/umdvegparam.txt"
clmdrv%CLM2_CHTFILE = "input/BCS/clm_parms/clm2_ptcanhts.txt"
clmdrv%CLM2_ISM = 0.45
clmdrv%CLM2_IT = 290.0
clmdrv%CLM2_ISCV = 0.
/

&noah
noahdrv%WRITEINTN = 3
noahdrv%NOAH_RFILE = "noah.rst"
noahdrv%NOAH_MGFILE = "input/BCS/1_4deg/GFRAC/"
noahdrv%NOAH_ALBFILE = "input/BCS/1_4deg/ALBEDO/"
noahdrv%NOAH_VFILE = "input/BCS/noah_parms/noah.vegparms.txt"
noahdrv%NOAH_SFILE = "input/BCS/noah_parms/noah.soilparms.txt"
noahdrv%NOAH_MXSNAL = "input/BCS/1_4deg/ALBEDO/maxsnalb.1gd4r"
noahdrv%NOAH_TBOT = "input/BCS/1_4deg/NOAH/tbot_0.25.1gd4r"
noahdrv%NOAH_ISM = 0.30
noahdrv%NOAH_IT = 290.0
noahdrv%NOAH_NVEGP = 7
noahdrv%NOAH_NSILP = 10
noahdrv%albedo_type = 1
noahdrv%gfrac_type = 1
noahdrv%mxsnalb_type = 1
noahdrv%tbot_type = 1
/

&vic
vicdrv%WRITEINTVIC = 3
vicdrv%VIC_NLAYER = 3
vicdrv%VIC_NNODE = 5
vicdrv%VIC_SNOWBAND = 1
vicdrv%VIC_ROOTZONES = 2
vicdrv%vic_frozen_soil = 1
vicdrv%VIC_SFILE = "input/BCS/vic_parms/soil.txt"
vicdrv%VIC_VEGLIBFILE = "input/BCS/vic_parms/veg_lib.txt"
vicdrv%VIC_RFILE = "restart.dat"
vicdrv%VIC_DSMAFFILE = "input/BCS/1_4deg/VIC/ds.1gd4r"
vicdrv%VIC_DSMAFFILE = "input/BCS/1_4deg/VIC/dsmax.1gd4r"
vicdrv%VIC_WSMAPFILE = "input/BCS/1_4deg/VIC/ws.1gd4r"

```

```

vicdrv%VIC_INFILTMAPFILE      = "input/BCS/1_4deg/VIC/infilt.1gd4r"
vicdrv%VIC_DEPTH1MAPFILE     = "input/BCS/1_4deg/VIC/depth1.1gd4r"
vicdrv%VIC_DEPTH2MAPFILE     = "input/BCS/1_4deg/VIC/depth2.1gd4r"
vicdrv%VIC_DEPTH3MAPFILE     = "input/BCS/1_4deg/VIC/depth3.1gd4r"
vicdrv%vic_initial_surf_temp = 290.0
/

```

```

&mos
mosdrv%WRITEINTM      = 24
mosdrv%MOS_RFILE      = "mos.rst"
mosdrv%MOS_VFILE      = "input/BCS/mos_parms/vegiparms_igbp.txt"
mosdrv%MOS_MVFILE     = "input/BCS/mos_parms/monvegpar_igbp.txt"
mosdrv%MOS_SFILE      = "input/BCS/mos_parms/real.soilparms.txt"
mosdrv%MOS_KVFILE     = "input/BCS/mos_parms/real.vegiparms.randy.txt"
mosdrv%MOS_KMVFILE    = "input/BCS/mos_parms/real.monvegpar.randy.txt"
mosdrv%MOS_KSFILE     = "input/BCS/mos_parms/real.soilparms.randy.txt"
mosdrv%MOS_POFILE     = "gswp2data/Fixed/W_sat.nc"
mosdrv%MOS_SIFILE     = "gswp2data/Fixed/Silt.nc"
mosdrv%MOS_SLFILE     = "gswp2data/Fixed/Slope.nc"
mosdrv%MOS_ISM        = 0.75
mosdrv%MOS_IT         = 290.0
mosdrv%MOS_IC         = 1
mosdrv%MOS_SMDA       = 0
mosdrv%MOS_TDA        = 0
mosdrv%MOS_SDA        = 0
mosdrv%MOS_NVEGP      = 24
mosdrv%MOS_NMVEGP     = 6
mosdrv%MOS_NSILP      = 10
/

```

```

&hyssib
hyssibdrv%WRITEINT    = 24
hyssibdrv%STATEVAR_AVG = 1
hyssibdrv%HYSSIB_RFILE = "hyssib.rst"
hyssibdrv%HYSSIB_ALBFILE = "/X8RAID/HYSSIB/"
hyssibdrv%HYSSIB_GALBRES = "1.0"
hyssibdrv%HYSSIB_AFILE = "/X8RAID/hyssib_parms/ssibalb"
hyssibdrv%HYSSIB_VFILE = "/X8RAID/hyssib_parms/ssibveg"
hyssibdrv%HYSSIB_TBOT = "/X8RAID/GSWP/NOAH/TBOT_GSWP2.BIN"
hyssibdrv%HYSSIB_TOPOSTD = "/X8RAID/HYSSIB/topostd_1.0.bfsa"
hyssibdrv%HYSSIB_ISM   = 0.75
hyssibdrv%HYSSIB_IT    = 290.0
hyssibdrv%HYSSIB_NVEGP = 20
hyssibdrv%HYSSIB_NVEGIP = 11
/

```

```
&ssib
ssibdrv%WRITEINT      = 24
ssibdrv%STATEVAR_AVG = 1
ssibdrv%SSIB_RFILE    = "ssib.rst"
ssibdrv%SSIB_ISM      = 0.30
ssibdrv%SSIB_IT       = 290.0
ssibdrv%SSIB_NVEGP    = 57
ssibdrv%SSIB_NVEGIP   = 12
ssibdrv%SSIB_FLGRES   = 0
/

&opendap
opendap_data_prefix = "/your/top-level/directory/here"
/
```

B Makefile

```
# Set up special characters

null :=
space := $(null) $(null)
doctool := ../../utils/docsgen.sh

# Check for directory in which to put executable
ifeq ($(MODEL_EXEDIR),$(null))
MODEL_EXEDIR := .
endif

# Check for name of executable
ifeq ($(EXENAME),$(null))
EXENAME := LIS
endif

# Check if SPMD is defined in "misc.h"
# Ensure that it is defined and not just "undef SPMD" set in file
ifeq ($(SPMD),$(null))
    SPMDSET := $(shell /bin/grep SPMD misc.h)
    ifneq (,$(findstring define,$(SPMDSET)))
        SPMD := TRUE
    else
        SPMD := FALSE
    endif
endif

# Determine platform
UNAMES := $(shell uname -s)
UMACHINE := $(shell uname -m)

ifeq ($(UNAMES),IRIX64)
INC_NETCDF := /u/jvg/local/netcdf-3.5.1/include
LIB_NETCDF := /u/jvg/local/netcdf-3.5.1/lib
ESMF_DIR := ../lib/esmf
LIB_ESMF := $(ESMF_DIR)/lib/lib0
MOD_ESMF := $(ESMF_DIR)/mod/mod0
endif

ifeq ($(UNAMES),AIX)

LIB_NETCDF := /usrx/local/netcdf
```

```

INC_NETCDF := /usrx/local/netcdf/include
LIB_MPI    := /usr/lpp/ppe.poe/lib
INC_MPI    := /usr/lpp/ppe.poe/include
ESMF_DIR   := /nfsuser/g01/wx20je/LDAS/src-par/esmf
LIB_ESMF   := $(ESMF_DIR)/lib/lib0
MOD_ESMF   := $(ESMF_DIR)/mod/mod0

endif

ifeq ($(UNAMES),OSF1)

LIB_MPI := /usr/lib
INC_MPI := /usr/include
ESMF_DIR := ../lib/esmf
LIB_ESMF := $(ESMF_DIR)/lib/lib0
MOD_ESMF := $(ESMF_DIR)/mod/mod0

endif

ifeq ($(UMACHINE), i686)

INC_NETCDF := /data1/netcdf/include
LIB_NETCDF := /data1/netcdf/lib
MPI_PREFIX := /data1/jim/local/mpich-1.2.4-absoft
LIB_MPI     := $(MPI_PREFIX)/lib
INC_MPI     := $(MPI_PREFIX)/include
ESMF_DIR    := ../lib/esmf
LIB_ESMF    := $(ESMF_DIR)/lib/lib0
MOD_ESMF    := $(ESMF_DIR)/mod/mod0

endif

ifeq ($(UMACHINE), ia64)

INC_NETCDF := /data1/netcdf/include
LIB_NETCDF := /data1/netcdf/lib
MPI_PREFIX := /opt/sgi/mpt/1.9.1.0
#MPI_PREFIX := /opt/sgi/mpt/1.11.0.0
LIB_MPI     := $(MPI_PREFIX)/lib
INC_MPI     := $(MPI_PREFIX)/include
ESMF_DIR    := ../lib/esmf
LIB_ESMF    := $(ESMF_DIR)/lib/lib0
MOD_ESMF    := $(ESMF_DIR)/mod/mod0

endif

```

```

# Load dependency search path.
dirs := . $(shell cat Filepath)
# Set cpp search path, include netcdf
cpp_dirs := $(dirs) $(INC_NETCDF) $(INC_MPI)
cpp_path := $(foreach dir,$(cpp_dirs),-I$(dir)) # format for command line

# Expand any tildes in directory names. Change spaces to colons.
VPATH := $(foreach dir,$(cpp_dirs),$(wildcard $(dir)))
VPATH := $(subst $(space),:,$(VPATH))

#-----
# Primary target: build the model
#-----
all: $(MODEL_EXEDIR)/$(EXENAME)

# Get list of files and determine objects and dependency files
FIND_FILES = $(wildcard $(dir)/*.F $(dir)/*.f $(dir)/*.F90 $(dir)/*.c)
FILES      = $(foreach dir, $(dirs),$(FIND_FILES))
SOURCES    := $(sort $(notdir $(FILES)))
DEPS       := $(addsuffix .d, $(basename $(SOURCES)))
OBJS       := $(addsuffix .o, $(basename $(SOURCES)))
DOCS       := $(addsuffix .tex, $(basename $(SOURCES)))

$(MODEL_EXEDIR)/$(EXENAME): $(OBJS)
$(FC) -o $@ $(OBJS) $(FOPTS) $(LDFLAGS)
debug: $(OBJS)
        echo "FFLAGS: $(FFLAGS)"
        echo "LDFLAGS: $(LDFLAGS)"
        echo "OBJS: $(OBJS)"

#####
##### Architecture-specific flags and rules#####
#####

#-----
# SGI
#-----

ifeq ($(UNAMES),IRIX64)

ESMF_ARCH = IRIX64
FC         := f90
CPP        := /lib/cpp

# Library directories

```

```

LIB_DIR = ../lib/sgi-64/
HDFLIBDIR = $(LIB_DIR)
GFIOLIBDIR = $(LIB_DIR)
CPPFLAGS := -P
PSASINC :=

# non-opensap
CFLAGS := $(cpp_path) -64 -c -O2 -OPT:Olimit=0 -static -DIRIX64
FFLAGS = $(cpp_path) -64 -r4 -i4 -c -cpp -I$(MOD_ESMF)/$(ESMF_ARCH) \
-DHIDE_SHR_MSG -DNO_SHR_VMATH -DIRIX64 -O2 -OPT:Olimit=0 \
-static

# opensap
#CFLAGS := $(cpp_path) -64 -c -O2 -OPT:Olimit=0 -static -DIRIX64 -DOPENSAP
#FFLAGS = $(cpp_path) -64 -r4 -i4 -c -cpp -I$(MOD_ESMF)/$(ESMF_ARCH) \
-DHIDE_SHR_MSG -DNO_SHR_VMATH -DIRIX64 -O2 -OPT:Olimit=0 \
-static -DOPENSAP

# debugging
#CFLAGS := $(cpp_path) -64 -c -g -static -DIRIX64
#FFLAGS = $(cpp_path) -64 -r4 -i4 -c -cpp -I$(MOD_ESMF)/$(ESMF_ARCH) \
-DHIDE_SHR_MSG -DNO_SHR_VMATH -DIRIX64 -g -static

FOPTS = $(LIB_DIR)bacio_64_sgi $(LIB_DIR)w3lib_64_sgi

LDFLAGS = -64 -L$(LIB_ESMF)/$(ESMF_ARCH) -lesmf -lmpi -L$(LIB_NETCDF) \
-lnetcdf

# WARNING: -mp and -g together cause wrong answers

# WARNING: - Don't run hybrid on SGI (that's what the == -mp is all about)

ifeq ($(SPMD),TRUE)
    FFLAGS += -mp
    FFLAGS += -macro_expand
#    FFLAGS += -I$(INC_MPI) -macro_expand

#    LDFLAGS += -L$(LIB_MPI) -lmpi
else
    FFLAGS += -DHIDE_MPI
endif

.SUFFIXES:
.SUFFIXES: .F90 .c .o

.F90.o:

```

```

$(FC) $(FFLAGS) $<
.c.o:
cc $(cpp_path) $(CFLAGS) $<

endif

#-----
# AIX
#-----

ifeq ($(UNAMES),AIX)

ESMF_ARCH      = rs6000_64
#FC             = mpixlf90_r
FC             = mpixlf90
CPP            = /lib/cpp
#CC            = mpcc_r
CC            = mpcc
MPI_PATH       = /usr/lpp/ppe.poe

# Library directories
LIB_DIR        = /nwprod/lib/
LIB_DIR2       = ../lib/
HDFLIBDIR      = $(LIB_DIR)
GFIOLIBDIR     = $(LIB_DIR)
CPPFLAGS       = -P
PSASINC        =
CFLAGS         = $(cpp_path) -c -w -O -q64 -qcpluscmt
FFLAGS         = $(cpp_path) -I$(MOD_ESMF)/$(ESMF_ARCH) -I$(MPI_PATH)/include \
               -O -qstrict -qtune=auto -qarch=auto -qmaxmem=-1 -NS2000 \
               -qsave -c -q64 -DAIX
FOPTS          = $(LIB_DIR)libw3_4.a
LDFLAGS        = -bmap:map -bloadmap:lm -lmass -L$(LIB_ESMF)/$(ESMF_ARCH) \
               -lesmf -L$(LIB_DIR) -lbacio_4

ifeq ($(SPMD),TRUE)
# FFLAGS -= -mp
# FFLAGS += -macro_expand
# FFLAGS += -I$(INC_MPI) -macro_expand

# LDFLAGS += -L$(LIB_MPI) -lmpi
else
FFLAGS += -DHIDE_MPI
endif
.F.o:
$(FC) $(FFLAGS) $<
.F90.o:

```



```

$(FC) $(FFLAGS) $<
.f.o:
$(FC) $(FFLAGS) $<
.f90.o:
.c.o:
$(CC) $(cpp_path) $(CFLAGS) $<

endif

#-----
# Compaq alpha - Halem cluster
#-----
ifeq ($(UNAMES),OSF1)

ESMF_ARCH = alpha
FC         := f90
CPP        := /lib/cpp

# Library directories
LIB_DIR = ../lib/alpha-32/
CPPFLAGS := -P
PSASINC  :=
CFLAGS   := $(cpp_path) -n32 -DOSF1
FFLAGS   = $(cpp_path) -c -cpp -automatic -convert big_endian \
            -assume byterecl -arch ev6 -tune ev6 -fpe3\
            -I$(MOD_ESMF)/$(ESMF_ARCH) -DOSF1 \
            -DHIDE_SHR_MSG -DNO_SHR_VMATH
FFLAGS_DOTF90 = -DHIDE_SHR_MSG -DOSF1 -free -fpe3 -DNO_SHR_VMATH
FFLAGS_DOTF = -extend_source -omp -automatic
FOPTS = $(LIB_DIR)bacio_32_alpha $(LIB_DIR)w3lib_32_alpha
LDFFLAGS = -L$(LIB_ESMF)/$(ESMF_ARCH) -lesmf -lmpi
# WARNING: -mp and -g together cause wrong answers

# WARNING: - Don't run hybrid on SGI (that's what the -= -mp is all about)

ifeq ($(SPMD),TRUE)
    FFLAGS -= -mp
    FFLAGS += -I$(INC_MPI) -macro_expand

# LDFFLAGS += -L$(LIB_MPI) -lmpi
else
    FFLAGS += -DHIDE_MPI
endif

.SUFFIXES:
.SUFFIXES: .f .f90 .F90 .c .o

```

```

.f.o:
$(FC) $(FFLAGS) $<
.F90.o:
$(FC) $(FFLAGS) $<
.c.o:
cc -c $(cpp_path) $(CFLAGS) $<
.f90.o:
$(FC) $(FFLAGS) $<

endif
#-----
# Linux
#-----

ifeq ($(UMACHINE),i686)
ESMF_ARCH = linux_absoft

ifeq ($(ESMF_ARCH),linux_pgi)

FC          := $(MPI_PREFIX)/bin/mpif90
CC          := $(MPI_PREFIX)/bin/mpicc
#FC         := pgf90
CPP         := /lib/cpp

CFLAGS      := $(cpp_path) -c -O2 -DPGI
FFLAGS      := $(cpp_path) -c -O2 -DHIDE_SHR_MSG -DNO_SHR_VMATH -DPGI \
               -Mbyteswapio -r4 -i4 -Mpreprocess

LDFLAGS     := -lmpich -lm -Mlfs

endif
ifeq ($(ESMF_ARCH),linux_ifc)

#FC         := $(MPI_PREFIX)/bin/mpif90
FC          := /opt/intel_fc_80/bin/ifort
CPP         := /lib/cpp

CFLAGS      := $(cpp_path) -c -O2 -DIFC

#FFLAGS     = $(cpp_path) -c -I$(MOD_ESMF)/$(ESMF_ARCH) -DHIDE_SHR_MSG \
               -DNO_SHR_VMATH -O

FFLAGS      = $(cpp_path) -c -O -nomixed_str_len_arg -names lowercase \
               -convert big_endian -assume byterecl -DHIDE_SHR_MSG \

```

```

-DNO_SHR_VMATH -DIFC

#LDFLAGS      = -L$(LIB_ESMF)/$(ESMF_ARCH) -lesmf -lmpich
LDFLAGS      =

endif
ifeq ($(ESMF_ARCH),linux_absoft)

#FC           := $(MPI_PREFIX)/bin/mpif90
#CC           := $(MPI_PREFIX)/bin/mpicc
#FC           := f90
#CC           := gcc
#CPP          := /lib/cpp

# non-openssl, no netcdf, no esmf
CFLAGS       := $(cpp_path) -c -O2 -Wall -DABSOFT -DLITTLE_ENDIAN
FFLAGS       := $(cpp_path) -c -O2 -YEXT_NAMES=LCS -s -B108 -YCFRL=1 \
               -YDEALLOC=ALL -DHIDE_SHR_MSG -DNO_SHR_VMATH -DABSOFT \
               -DLITTLE_ENDIAN

# non-openssl, no esmf
#CFLAGS       := $(cpp_path) -c -O2 -Wall -DABSOFT -DLITTLE_ENDIAN
#FFLAGS       := $(cpp_path) -c -O2 -YEXT_NAMES=LCS -s -B108 -YCFRL=1 \
               -YDEALLOC=ALL -p$(INC_NETCDF) -DHIDE_SHR_MSG -DNO_SHR_VMATH \
               -DABSOFT -DLITTLE_ENDIAN

# non-openssl
#CFLAGS       := $(cpp_path) -c -O2 -Wall -DABSOFT -DLITTLE_ENDIAN
#FFLAGS       := $(cpp_path) -c -O2 -YEXT_NAMES=LCS -s -B108 -YCFRL=1 \
               -YDEALLOC=ALL -p$(MOD_ESMF)/$(ESMF_ARCH) -p$(INC_NETCDF) \
               -DHIDE_SHR_MSG -DNO_SHR_VMATH -DABSOFT -DLITTLE_ENDIAN

# openssl, no netcdf, no esmf
#CFLAGS       := $(cpp_path) -c -O2 -Wall -DABSOFT -DLITTLE_ENDIAN -DOPENSSL
#FFLAGS       := $(cpp_path) -c -O2 -YEXT_NAMES=LCS -s -B108 -YCFRL=1 \
               -YDEALLOC=ALL -DHIDE_SHR_MSG -DNO_SHR_VMATH -DABSOFT \
               -DLITTLE_ENDIAN -DOPENSSL

# openssl, farmer-dog-bones, no netcdf, no esmf
#CFLAGS       := $(cpp_path) -c -O2 -Wall -DABSOFT -DLITTLE_ENDIAN -DOPENSSL \
               -DFARMER_DOG_BONES
#FFLAGS       := $(cpp_path) -c -O2 -YEXT_NAMES=LCS -s -B108 -YCFRL=1 \
               -YDEALLOC=ALL -DHIDE_SHR_MSG -DNO_SHR_VMATH -DABSOFT \
               -DLITTLE_ENDIAN -DOPENSSL -DFARMER_DOG_BONES

# openssl

```

```

#CFLAGS      := $(cpp_path) -c -O2 -Wall -DABSOFT -DLITTLE_ENDIAN -DOPENDAP
#FFLAGS      := $(cpp_path) -c -O2 -YEXT_NAMES=LCS -s -B108 -YCFRL=1 \
               -YDEALLOC=ALL -p$(MOD_ESMF)/$(ESMF_ARCH) -p$(INC_NETCDF) \
               -DHIDE_SHR_MSG -DNO_SHR_VMATH -DABSOFT -DLITTLE_ENDIAN \
               -DOPENDAP

# debugging with opendap
#CFLAGS      := $(cpp_path) -c -g -DABSOFT -DLITTLE_ENDIAN -DOPENDAP
#FFLAGS      := $(cpp_path) -c -O1 -g -Rb -Rc -Rs -Rp -YEXT_NAMES=LCS -s \
               -B108 -YCFRL=1 -YDEALLOC=ALL -p$(MOD_ESMF)/$(ESMF_ARCH) \
               -p$(INC_NETCDF) -DHIDE_SHR_MSG -DNO_SHR_VMATH -DABSOFT \
               -DLITTLE_ENDIAN -DOPENDAP

# debugging without opendap
#CFLAGS      := $(cpp_path) -c -g -DABSOFT -DLITTLE_ENDIAN
#FFLAGS      := $(cpp_path) -c -O1 -g -Rb -Rc -Rs -Rp -YEXT_NAMES=LCS -s \
               -B108 -YCFRL=1 -YDEALLOC=ALL -p$(MOD_ESMF)/$(ESMF_ARCH) \
               -p$(INC_NETCDF) -DHIDE_SHR_MSG -DNO_SHR_VMATH -DABSOFT \
               -DLITTLE_ENDIAN

# debugging without opendap, esmf, netcdf
##CFLAGS      := $(cpp_path) -c -gdwarf -DABSOFT -DLITTLE_ENDIAN
#CFLAGS      := $(cpp_path) -c -g -DABSOFT -DLITTLE_ENDIAN
#FFLAGS      := $(cpp_path) -c -O1 -g -m0 -Rb -Rc -Rs -Rp -YEXT_NAMES=LCS -s \
               -B108 -YCFRL=1 -YDEALLOC=ALL -DHIDE_SHR_MSG -DNO_SHR_VMATH \
               -DABSOFT -DLITTLE_ENDIAN

# profiling -- don't forget the -P in LDFLAGS
#CFLAGS      := $(cpp_path) -c -O2 -DABSOFT -DLITTLE_ENDIAN -DOPENDAP -pg
#FFLAGS      := $(cpp_path) -c -O2 -YEXT_NAMES=LCS -s -B108 -YCFRL=1 \
               -YDEALLOC=ALL -p$(MOD_ESMF)/$(ESMF_ARCH) -p$(INC_NETCDF) \
               -DHIDE_SHR_MSG -DNO_SHR_VMATH -DABSOFT -DLITTLE_ENDIAN \
               -DOPENDAP -P

#LDFLAGS      := -L$(LIB_ESMF)/$(ESMF_ARCH) -L$(LIB_NETCDF) -lnetcdf -lesmf \
               -lmpich -lU77 -lm
#LDFLAGS      := -L$(LIB_NETCDF) -lnetcdf -lU77 -lm
#LDFLAGS      := -lmpich -lU77 -lm
LDFLAGS      := -lU77 -lm
endif

ifeq ($(ESMF_ARCH),linux_lf95)

#FC           := $(MPI_PREFIX)/bin/mpif90
FC           := lf95
CPP          := /lib/cpp

```

```

CFLAGS      := $(cpp_path) -c -O -DUSE_GCC -DLAHEY -DLITTLE_ENDIAN
FFLAGS      := $(cpp_path) -c -O -DHIDE_SHR_MSG -DLINUX -DNO_SHR_VMATH \
              -I$(MOD_ESMF)/$(ESMF_ARCH) -DLAHEY -DLITTLE_ENDIAN
#LDFLAGS     := -L$(LIB_ESMF)/$(ESMF_ARCH) -lesmf -L$(LIB_MPI) -lmpich -s \
              --staticlink
#LDFLAGS     := -s --staticlink
LDFLAGS     := -s

endif

# Library directories
#LIB_DIR    = ../lib/pc-32/$(ESMF_ARCH)/
LIB_DIR     = ../w3lib/
HDFLIBDIR   = $(LIB_DIR)
GFIOLIBDIR  = $(LIB_DIR)
CPPFLAGS    := -P
PSASINC     :=
#FOPTS      = $(LIB_DIR)bacio_32_pclinux $(LIB_DIR)w3lib_32_pclinux
FOPTS       = $(LIB_DIR)libw3.a
# WARNING: -mp and -g together cause wrong answers

# WARNING: - Don't run hybrid on SGI (that's what the -= -mp is all about)

ifeq ($(SPMD),TRUE)
# FFLAGS  -= -mp
# FFLAGS  += -macro_expand
# FFLAGS  += -I$(INC_MPI) -macro_expand

# LDFLAGS += -L$(LIB_MPI) -lmpi
else
  FFLAGS += -DHIDE_MPI
endif

.SUFFIXES:
.SUFFIXES: .F90 .c .o

.F90.o:
$(FC) $(FFLAGS) $<
.c.o:
$(CC) $(cpp_path) $(CFLAGS) $<

endif

ifeq ($(UMACHINE),ia64)
FC      := efc

```

```

#FC          := ifort
#FC          := $(MPI_PREFIX)/bin/mpif90
#CC          := $(MPI_PREFIX)/bin/mpicc
CPP          := /lib/cpp

CFLAGS       := $(cpp_path) -c -O2 -I $(INC_MPI) -DIFC
FFLAGS       := $(cpp_path) -c -O -DHIDE_SHR_MSG -DNO_SHR_VMATH -DIFC

#CFLAGS      := $(cpp_path) -c -O2 -DIFC -I $(INC_MPI)
#FFLAGS      := $(cpp_path) -c -O -nomixed_str_len_arg -names lowercase \
               -convert big_endian -assume byterecl -DHIDE_SHR_MSG \
               -DNO_SHR_VMATH -DIFC

#LDFLAGS     := -Vaxlib -lmpi
LDFLAGS      := -Vaxlib

# Library directories
#LIB_DIR     = ../lib/pc-32/$(ESMF_ARCH)/
LIB_DIR      = ../w3lib/
HDFLIBDIR    = $(LIB_DIR)
GFIOLIBDIR   = $(LIB_DIR)
CPPFLAGS     := -P
PSASINC      :=
#FOPTS       = $(LIB_DIR)bacio_32_pclinux $(LIB_DIR)w3lib_32_pclinux
FOPTS        = $(LIB_DIR)libw3.a
# WARNING: -mp and -g together cause wrong answers

# WARNING: - Don't run hybrid on SGI (that's what the == -mp is all about)

ifeq ($(SPMD),TRUE)
# FFLAGS    -= -mp
# FFLAGS    += -macro_expand
# FFLAGS    += -I$(INC_MPI) -macro_expand

# LDFLAGS   += -L$(LIB_MPI) -lmpi
else
    FFLAGS   += -DHIDE_MPI
endif

.SUFFIXES:
.SUFFIXES: .F90 .c .o

.F90.o:
$(FC) $(FFLAGS) $<
.c.o:
$(CC) $(cpp_path) $(CFLAGS) $<

```

```

endif

RM := rm
# Add user defined compiler flags if set, and replace FC if USER option set.
FFLAGS += $(USER_FFLAGS)
ifneq ($(USER_FC),$(null))
FC := $(USER_FC)
endif

clean:
$(RM) -f *.o *.mod *.stb $(MODEL_EXEDIR)/$(EXENAME)

realclean:
$(RM) -f *.o *.d *.mod *.stb $(MODEL_EXEDIR)/$(EXENAME)
doc:
$(doctool)
#-----
#!!!!!!!!!!!!!!!!!!!!DO NOT EDIT BELOW THIS LINE!!!!!!!!!!!!!!!!!!!!
#-----
# These rules cause a dependency file to be generated for each source
# file. It is assumed that the tool "makdep" (provided with this
# distribution in clm2/tools/makdep) has been built and is available in
# the user's $PATH. Files contained in the clm2 distribution are the
# only files which are considered in generating each dependency. The
# following filters are applied to exclude any files which are not in
# the distribution (e.g. system header files like stdio.h).
#
# 1) Remove full paths from dependencies. This means gnumake will not break
#    if new versions of files are created in the directory hierarchy
#    specified by VPATH.
#
# 2) Because of 1) above, remove any file dependencies for files not in the
#    clm2 source distribution.
#
# Finally, add the dependency file as a target of the dependency rules. This
# is done so that the dependency file will automatically be regenerated
# when necessary.
#
#    i.e. change rule
#    make.o : make.c make.h
#    to:
#    make.o make.d : make.c make.h
#-----
DEPGEN := ./MAKDEP/makdep -s F

```

```

%.d : %.c
@echo "Building dependency file $@"
@$(DEPGEN) -f $(cpp_path) $< > $@
%.d : %.f
@echo "Building dependency file $@"
@$(DEPGEN) -f $(cpp_path) $< > $@
%.d : %.F90
@echo "Building dependency file $@"
@$(DEPGEN) -f $(cpp_path) $< > $@
%.d : %.F
@echo "Building dependency file $@"
@$(DEPGEN) -f $(cpp_path) $< > $@
#
# if goal is clean or realclean then don't include .d files
# without this is a hack, missing dependency files will be created
# and then deleted as part of the cleaning process
#
INCLUDE_DEPS=TRUE
ifeq ($(MAKECMDGOALS), realclean)
    INCLUDE_DEPS=FALSE
endif
ifeq ($(MAKECMDGOALS), clean)
    INCLUDE_DEPS=FALSE
endif

ifeq ($(INCLUDE_DEPS), TRUE)
    -include $(DEPS)
endif

```


References

- [1] GrADS. <http://grads.iges.org/grads/grads.html>.
- [2] Protex documenting system. <http://gmao.gsfc.nasa.gov/software/protex>.
- [3] ALMA. <http://www.lmd.jussieu.fr/ALMA/>.
- [4] CLM. <http://www.cgd.ucar.edu/tss/clm>.
- [5] DODS. <http://www.unidata.ucar.edu/packages/dods/>.
- [6] NOAH. <ftp://ftp.ncep.noaa.gov/pub/gcp/ldas/noahlsn/>.
- [7] VIC. <http://hydrology.princeton.edu/research/lis/index.html>.
- [8] W3FI63 PROGRAM. <http://dss.ucar.edu/datasets/ds609.1/software/mords/w3fi63.f>.
- [9] G. J. Collatz, C. Grivet, J. T. Ball, and J. A. Berry. Physiological and environmental regulation of stomatal conductance: Photosynthesis and transpiration: A model that includes a laminar boundary layer. *Agric. For. Meteorol.*, 5:107–136, 1991.
- [10] Chen. F., Mitchell. K., Schaake. J, Xue. J, Pan. H, Koren. V., Ek. M Duan, and A. Betts. Modeling of land-surface evaporation by four schemes and comparison with fife observations. *J. Geophys. Res.*, 101(D3):7251–7268, 1996.
- [11] P. G. Jarvis. The interpretation of leaf water potential and stomatal conductance found in canopies of the field. *Phil. Trans. R. Soc.*, 273:593–610, 1976.
- [12] L. A. Richards. Capillary conduction of liquids in porous media. *Physics*, 1:318–333, 1931.
- [13] E. Rogers, T. L. Black, D. G. Deaven, G. J. DiMego, Q. Zhao, M. Baldwin, N. W. Junker, and Y. Lin. Changes to the operational “early” eta analysis/forecast system at the national centers of environmental prediction. *Wea. Forecasting*, 11:391–413, 1996.